

NASA Contractor Report 182069

A Real Time Dynamic Data Acquisition and Processing System for Velocity, Density, and Total Temperature Fluctuation Measurements

Steven J. Clukey

Vigyan , Inc.

Hampton, VA 23666

Contract NAS1-18585

FEBRUARY 1991

(NASA-CR-182069) A REAL TIME DYNAMIC DATA
ACQUISITION AND PROCESSING SYSTEM FOR
VELOCITY, DENSITY, AND TOTAL TEMPERATURE
FLUCTUATION MEASUREMENTS (Vigyan Research
Associates) 278 p

NO1-18094

Unclass

CSCL 01C 03/03 0000159



National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

CONTENTS

Abstract	1
Symbols	2
1. INTRODUCTION	3
2. SYSTEM DESCRIPTION	5
2.1. HARDWARE	5
2.1.1. Amplifier and Filter Subsystem	5
2.1.2. High Speed Digitizer Subsystem	6
2.1.3. Low speed digitizer	6
2.1.4. DDAPS link to tunnel computer	6
2.1.5. Auxiliary interfaces	8
2.1.5.1. FM Tape/Time Code Generator Interface	8
2.1.5.2. NEFF 130 Amplifier Interface	8
2.1.6. Computer Peripherals	9
2.1.6.1. Disk storage	9
2.1.6.2. Tape drives	9
2.1.6.3. Display	9
2.1.6.4. Plotter	9
2.1.6.5. Printer	10
2.1.7. Array Processor	10
2.2. SOFTWARE	11
2.2.1. Software environment	11
2.2.2. Baseline software	11
2.2.3. Hot wire application software	12
2.2.3.1. MODUSR1	12
2.2.3.2. MODUSR2	13
2.2.3.3. MODUSR3	13
2.2.3.4. MODUSR4	14
2.2.3.5. MODUSR5	14
2.2.3.6. MODUSR6	15
2.2.3.7. MODUSR8	15
2.2.3.8. MOD7	15
3. OPERATION	17
3.1. System setup - hardware	17

3.2. System setup - software	18
3.3. ACQUIRE installation	20
3.4. Tailoring the test setup	20
3.5. Getting started	23
3.6. Sequence program - initialization	24
3.7. Sequence program - acquisition	25
3.8. File transfer to PC	26
3.8.1. LOGFILE TO PC	26
3.8.2. 3.8.3. File conversion in DOS	26
4. SYSTEM THEORY	28
4.1. HARDWARE	28
4.1.1. Data Flow	28
4.1.1.1. Mean Quantities	28
4.1.1.2. Fluctuating Quantities	29
4.1.1.2.1. Bandwidth	30
4.1.1.2.2. Aggregate Bandwidth	30
4.1.1.2.3. Data Buffering	30
4.1.2. FM Tape/Time Code Generator Interface	30
4.1.3. NEFF 130 Amplifier Interface	32
4.2. SOFTWARE	33
4.2.1. ACQUIRE	33
4.2.1.1. Configuration files	34
4.2.1.2. System variables	34
4.2.1.3. Binary switches	35
4.2.1.4. Plot setup	35
4.2.1.5. Sequence program files	36
4.2.2. APPLICATION ENHANCEMENTS	36
4.2.2.1. ACQUIRE Expansion	36
4.2.2.2. APPLICATION Additions	38
4.2.2.2.1. Hot wire data acquisition	38
4.2.2.2.2. Calibration	39
4.2.2.2.3. Log files	40
4.2.2.2.3. Coefficient Files	41
4.2.2.2.4. Dynamic data	42
4.2.2.2.5. Fluctuating data	42
4.2.2.2.6. Coefficient calculations	44
4.2.2.2.7. Sensitivity calculations	45
4.2.2.2.8. Calculating fluctuations	46
4.2.2.2.9. Precision filter system control	48
4.2.2.2.10. FM Tape Control	50
4.2.2.2.11. Scanner/Voltmeter Control	52

4.2.2.2.12. Data Packet Transmission to SDS	52
4.2.2.2.13. NEFF 130 Amplifier Gain Query	52
4.2.2.2.14. Data Packets	53
4.2.3. Utility Functions	54
4.2.3.1. File utilities	54
4.2.3.2. Scalar plot utilities	54
4.2.3.3. Process utilities	55
5. SOFTWARE SUPPORT (PROGRAM DEVELOPMENT)	56
5.1. Array processor program development	56
5.2. UNIX Operating System	57
6. SYSTEM SUMMARY	58
6.1. SOFTWARE	58
7. Planned Future Enhancements	60
7.1. Connect the digitizer to the array processor	61
7.2. DSP - Array Processor integration	61
7.3. Utilize UNIX operating system	62
7.4. Translate ACQUIRE code to Unix compatible languages	62
INDEX	277

APPENDIX

APPENDIX A. Function definitions	81
APPENDIX B. Processing Comparison Results	243
APPENDIX C. Sequence Programs	247
APPENDIX D. FM Tape and Time Code Generator Interface	265
APPENDIX E. NEFF 130 Amplifier Interface	269
APPENDIX F. Supporting and Related Documentation	275

FIGURES

Figure 1. System Block Diagram	65
Figure 2. Analog Cabling Diagram - data acquisition	66
Figure 3. Analog Cabling Diagram - tape playback	67
Figure 4. Plot: hot wire voltage vs. mass flow	68
Figure 5. Observation Report	69
Figure 6. Plot: Waveforms - Typical time trace	70
Figure 7. Bottleneck fix - a Proposal	71

TABLES

Table 1. Equipment list	73
Table 2. Packet Format for 8'TPT test	74
Table 3. Packet Format for 4" pipe flow test	75
Table 4. Fluctuating Data File Name Format (voltages)	76
Table 5. Fluctuating Data File Name Format (computed)	77

ACKNOWLEDGEMENTS and REFERENCES

Acknowledgements	63
References:	64

Abstract

This report describes the real time Dynamic Data Acquisition and Processing System (DDAPS) which provides a mechanism for the simultaneous measurement of velocity, density, and total temperature fluctuations. The system of hardware and software is described in context of the wind tunnel environment.

The DDAPS replaces both a recording mechanism and a separate data processing system. DDAPS receives input from hot wire anemometers. Amplifiers and filters condition the signals with computer controlled modules. The analog signals are simultaneously digitized and digitally recorded on disk. Automatic acquisition collects necessary calibration and environment data. Hot wire sensitivities are generated and applied to the data to compute fluctuating quantities. The presentation of the raw and processed data is accomplished on demand.

This paper describes the interface to DDAPS and the internal mechanisms of DDAPS. A summary of operations relevant to the use of the DDAPS is also provided. This report supercedes NASA Contractor Report 181758.

PRECEDING PAGE BLANK NOT FILMED

Symbols

A_1-A_8	Constants in equation (1)
E	mean voltage across wire
e'	wire voltage perturbation (total voltage - E)
G_w	amplification factor (instrumentation)
KB	Kilobyte (Kilo=1024, byte=8 binary bits)
MB	Megabyte (1024 Kilobytes)
m	mass flow
m'	mass flow perturbation
P_0	total pressure
p'	pressure perturbation
S_u	velocity sensitivity $\left(\frac{\partial \log E}{\partial \log U} \right)_{\rho, T_0, T_w}$
S_ρ	density sensitivity $\left(\frac{\partial \log E}{\partial \log \rho} \right)_{U, T_0, T_w}$
S_{T_0}	temperature sensitivity $\left(\frac{\partial \log U}{\partial \log T_0} \right)_{U, \rho, T_w}$
T_0	mean total temperature
T_0'	total temperature perturbation
T_w	mean temperature of heated wire
U	mean velocity
u'	velocity perturbation
ρ	mean density
ρ'	density perturbation

Superscripts

\sim	rms (root mean square)
$-$	mean
$'$	instantaneous value

1. INTRODUCTION

Recent implementation of a hot wire anemometry technique suggests that a three wire probe can separate three components of the perturbations in the flow field. Velocity, density and total temperature fluctuations may be determined using three hot wires, since at subsonic and transonic speeds it may be shown that the voltage measured across a heated wire mounted normal to the flow and operated with a constant temperature anemometer may be a function of velocity, density and total temperature^{1, 2}. Under these conditions, a single equation is selected for the fluctuating voltage across each wire which is assumed to be a function of the three variables - velocity, density, and total temperature. Quantitative measurements for the three fluctuations in the flow variables use probes with three wires mounted normal to the flow. An effort is made to separate the sensitivities of each wire from the others by using different diameter wires and different "overheats".

The development of a dedicated hardware and software system to support hot wire anemometry research at NASA Langley Research Center was precipitated by the necessity to process simultaneous hot wire data from three wire probes more rapidly than previously possible.

Prior to the development of DDAPS, all data was acquired on FM tape, and all processing was done in an off-line batch mode. This method delayed recognition of faulty or incomplete data, and test results were often delayed several months.

During a flow diagnostics test in the 8 Foot Transonic Pressure Tunnel (8'TPT) at NASA Langley Research Center in January of 1988^{3, 4}, the DDAPS was connected in parallel to the existing test instrumentation systems to provide an initial test bed for the new system. The DDAPS was not designated as a primary data acquisition and reduction system, but it soon

became apparent that the data logging capabilities would be especially helpful in collecting the hot wire calibration data in an easily manageable format. The hot wire calibration data and the generation of hot wire sensitivities were processed only by the DDAPS and the calibration data and sensitivities were used both by DDAPS and by other data processing facilities. As a test of the digitization and recording capability, dynamic data was routinely digitized in parallel with the FM tape recordings. As soon as adequate calibration data was collected, the DDAPS processed some of the data, and provided velocity, density, and total temperature turbulence measurements. These results compared favorably with subsequent off-line batch data processing (see Appendix B.).

A second test was supported to compare hot wire techniques and laser velocimetry techniques in the Basic Aerodynamic Research Facility (B.A.R.F.).³

A more recent test in a 4" pipe flow apparatus was supported in June, 1990 to compare three wire and single wire hot wire techniques.⁵ Pre-calibrated hot-wire probes were available. This test did not use any parallel system (FM tape), and provided real time data reduction.

The DDAPS provides the processes necessary to:

- 1) acquire the hot wire calibration data
- 2) acquire the dynamic hot wire data
- 3) generate hot wire coefficients and sensitivities
- 4) compute velocity, density, and total temperature fluctuations
- 5) compute other statistical relationships
- 6) provide spectral analysis
- 7) data management
- 8) produce data reports and plots

2. SYSTEM DESCRIPTION

DDAPS is a system of hardware and software based on systems purchased from Data Laboratories, Ltd., Precision Filters, Inc., Analogic, Inc., and Hewlett Packard Corporation. Modifications and enhancements to the software and hardware have converted a waveform recorder into a hot wire anemometry acquisition and processing system specifically tailored to a three wire technique that yields separate velocity, density and total temperature components of turbulence.

2.1. HARDWARE

The system (figures 1., 2., 3.) is divided into an analog front end, and a computer-based processing and display section. The analog front end consists of a filter/amplifier subsystem, a high speed digitizer, and a low speed digitizer (or a data link to another acquisition computer). An FM tape subsystem is available for companion recording and playback capability. All are fully computer controlled. The processing and display subsystem controls the analog subsystems, and receives the digitized data, processes the data, displays the data, and stores the data in a permanent file.

2.1.1. Amplifier and Filter Subsystem

The analog signals from the hot wire anemometers are first routed to the Precision Filters, Inc. precision amplifier and filter subsystem. This subsystem is currently configured for four channels, providing support for only one three wire probe. Each channel successively passes the anemometer signal through a pre-amplifier, high pass filter, low pass filter, and a post-amplifier. The full bandwidth capability of each channel is .1Hz to 100KHz (4 of 12 channels are extended to 200KHz). The

high pass filter acts as the anti-aliasing filter for the high speed digitizer.

2.1.2. High Speed Digitizer Subsystem

A high speed digitizer, called a Multitrap modular waveform recorder by Data Laboratories, Ltd., is configured to digitize up to 14 channels of fluctuating hot wire data (three channels are required for each 3-wire probe) at rates of up to one million (1M) samples per second (6 channels at up to 1M samples/sec, and 8 channels at up to 262144 (256K) samples/second. This data is stored temporarily in the Multitrap buffer memories (up to 256K samples per channel), and then transferred to the HP 9000/330 computer memory at 100,000 sample per second rate - one channel at a time - via a dedicated 16 bit parallel bus (GPIO).

2.1.3. Low speed digitizer

This subsystem is not currently implemented - it was never procured. An existing wind tunnel data acquisition system provides the functions of a low speed digitizer subsystem. However, the low speed digitizer subsystem would consist of a multiplexer and digitizer which would digitize mean values (0 Hz to 1 Hz) such as calibration data and tunnel parameter data, which would be logged for further processing of the hot-wire calibration and fluctuating data.

2.1.4. DDAPS link to tunnel computer

The IEEE-488 standard bus, or General Purpose Interface Bus (GPIB), is used to receive static data from an existing tunnel data acquisition system computer. The tunnel computer transmits a packet of ASCII data relevant to the tunnel conditions and hot wire calibration data. This

link was selected because of its general availability in various tunnel computers. It provides a minimum transfer rate of 82,000 byte per second, which is more than adequate to receive as many as four complete ASCII data packets per second.

The packet format used to support the 8' TPT test is shown in Table 3. The packet format used to support the 4" pipe flow test is shown in Table 2.

The packet format may vary from one tunnel computer to another, simply due to the implementation of a variant format..

The data items in the packet can vary, depending on the experimental variables provided by the tunnel computer. For example, if the experiment involves two three-wire probes, and the tunnel computer digitizes the the mean values of six wires, then six values would be in the packet. If the probe position is measured by the tunnel computer, then that information could also be transmitted to DDAPS.

Use of an existing tunnel data acquisition system to collect the tunnel conditions and the additional mean values related to the hot wire calibration data eliminates the need for a parallel hardware system (the low speed digitizer), and the need to develop instrument calibration software and hardware. However, it does require additional work for the tunnel computer personnel to configure their acquisition setup to handle the additional channels, and to provide the GPIB software to generate the data packets for the DDAPS.

The link is configured such that the DDAPS end is not the system controller, but as device 01. This was accomplished by setting switches on the HP 98624A GPIB Interface Card inserted into the computer specifically for the link. The select code is set to 8; interrupts are not relevant,

since they are not used. The tunnel computer providing the data packet is configured as system controller, and outputs ASCII data packets at a rate set by the tunnel computer.

The packet is read into a DDAPS packet buffer with one program statement in the subroutine Get_packet in module MODUSR2: ENTER Pkt_sc:Pkt\$(*), where Pkt_sc is equal to 8, and the Pkt\$ array was sized for 47 strings, each 80 characters long.

2.1.5. Auxiliary interfaces

Two special interfaces have been developed to connect hardware to DDAPS.

2.1.5.1. FM Tape/Time Code Generator Interface

The FM tape interface provides a connection to both the FM tape recorder, and the time code generator. This connection includes logic which enables the HP 98623A BCD I/O interface card to control the FM tape and the time code generator. The cables between this card and the FM tape also carry the BCD-coded time from the time code generator. Appendix D. details the hardware, and module MODUSR4 implements the software (see 2.2.3.4.).

2.1.5.2. NEFF 130 Amplifier Interface

The NEFF 130 interface provides a connection to 10 NEFF 130 amplifiers, so that the gain selected by front panel control can be read by DDAPS. Each amplifier provides a single BCD digit, containing the gain code. Appendix E. details the hardware, and module MODUSR5 implements the software (see 2.2.3.5.).

2.1.6. Computer Peripherals

2.1.6.1. Disk storage

A 20 MB, a 55 MB, and a 340 MB hard disk drive is available for programs. A 650 MB erasable optical disk drive - with removable disk pack - is available for test data. The 340 MB Winchester disk is designated for the future development of a UNIX-based DDAPS, and is currently supporting HP-UX. The 55 MB disk is designated for DDAPS development, and for statistics packages. The 20 MB disk is used for array processor development. In addition, a 1.2 MB $3\frac{1}{2}$ " floppy disk drive is available for program development and hot wire calibration data.

2.1.6.2. Tape drives

A 67 MB $1/4$ " streamer cartridge tape is available, and has been used for software installation and for archival of data. A $1/2$ ", 1600 bpi digital tape drive is also available, but is restricted to use under UNIX (not supported under BASIC).

2.1.6.3. Display

A color CRT is the system and data display console.

2.1.6.4. Plotter

An 8 pen autoloader flatbed plotter is available for plot generation, and is used to display dynamic data and hot wire calibration data.

2.1.6.5. Printer

A dot matrix printer is available for data display. It can produce screen dumps, but is used primarily to generate a record of the hot wire calibration data, and, as data processing is accomplished, the results of the processing are printed.

2.1.7. Array Processor

An array processor (AP), the AP509 from Analogic, Inc., provides a 90X performance enhancement when performing processing on time variant data vectors. This AP was selected for its compatibility with HP 9000/3XX computer I/O systems, its large vector memory size (1 M word), the availability of both BASIC and UNIX (Fortran) interface libraries, and its 9 MFLOP (floating point operations per second) performance. Its greatest asset is to solve the three wire simultaneous equation (see 4.2.2.2.8.), where compute times were reduced from 80 minutes to less than 1 minute. However, other functions are also similarly enhanced. These functions are implemented in routine MOD7.

2.2. SOFTWARE

2.2.1. Software environment

All DDAPS programs operate under the HP BASIC 5.0 operating system, in the interpretive BASIC language. Several compiled subroutines are part of the ACQUIRE software system to enhance computational speed in some of the software.

2.2.2. Baseline software

The ACQUIRE⁶ software system, version 1.3, provided by Data Laboratories, Ltd. is the basis for the Dynamic Data Acquisition and Processing System (DDAPS) used for the acquisition of hot wire anemometry data.

Since the ACQUIRE package is an off-the-shelf product, no attempt to describe its full capability will be made.

ACQUIRE has been modified in several minor ways to provide enhanced capabilities. These were necessary to support the application specific routines. The changes primarily affect internal control functions, but add to the sequence program repertoire as well. This allows fully automatic data acquisition and processing.

Routine MODSEQ implements the ACQUIRE sequence functions, and has been modified to provide the necessary mechanisms for a fully automated data acquisition and processing sequence.

The root program was modified by including the additional user common block */Usrcommunic/*, containing user status flag *Sysusr*. This common may be checked to determine if certain user functions are in progress - see

MODSEQ (function (WAIT !), MODUSR2 (function GET MEAN DATA), and MODUSR4 (functions FM AUTORECORD and FM AUTOPLAYBACK).

ACQUIRE routine MAIN1 was modified to separate the time-of-day clock setting function from the system initialization, and allows user selection of the SET TIME function.

2.2.3. Hot wire application software

Major additions were made to ACQUIRE in the form of sub-programs. These user written modules are configured according to guidelines provided by ACQUIRE, so they are automatically included in ACQUIRE. Additional sub-programs were written to support these major additions. The functions implemented in these sub-programs, or routines, are listed in Appendix A.. These routines utilize function call mechanisms and subroutines provided by ACQUIRE, as well as user application routines. Copies of both the source code and program listings are available from the author.

2.2.3.1. MODUSR1

Utility functions for scalar plotting, file maintenance, and system control are in this sub-program.

Scalar plots are defined in several functions whose name start with PLOTxxx... These functions plot data previously logged, and are pre formatted to plot specific data with specific axes, scales, labels, etc. These functions are often modified for particular test and reduction scenarios, and are therefore "dynamic code". Some examples of existing plots are incorporated in the available software.

File maintenance includes listing selected files (CATaloging with the SELECT option), copying selected files from one disk and/or subdirectory to another, moving selected files, and purging files and/or selected files from a specific disk and/or directory.

System control functions must be tailored to a selected test scenario and specific hardware configurations.

2.2.3.2. MODUSR2

Three wire hot-wire functions are implemented in this sub-program. These functions provide the required data management - logfile - and data processing capabilities. Chapter 4.2.2.2. describes this comprehensive sub-program.

2.2.3.3. MODUSR3

Precision filter control and status mechanisms are implemented by this sub-program to initialize, control and query current status of the amplifier/filter subsystem. The amplifier and filters are grouped into hot-wire channels, three in a group (actually, due to a concern for phase matching, one group has 2 channels, and another has 1 channel). This grouping allows front panel operation to change operational parameters for an entire three wire probe at a time. Operator lockout function PF REMOTE inhibits changes while data is being acquired. PF LOCAL re-enables front panel operation. Status query functions allow the data acquisition process access to current gain information on a per-channel basis. Hardware-software configuration compatibility is maintained in the support routine PF_INIT.

2.2.3.4. MODUSR4

Control of the FM tape subsystem is implemented using a custom interface connected to an HP 98623A BCD I/O interface card, a Honeywell 96 tape recorder, and a DATUM 9310 time code generator.

The use of any analog recording method is not a primary technique in DDAPS. However, since the FM tape is a traditional method of recording, and since the high speed digitizers can only buffer 256K samples, it is sometimes convenient to use FM tape. Tapes in the archive library must also be processed. Dynamic range of the FM tape is 52dB; Dynamic range of the high speed digitizer is 72dB.

This subsystem allows both automatic recording of analog data and automatic data reduction. The tape position is automatically recorded for a particular observation as a start and stop time from the time code generator, along with other data related to the observation. During data processing, the tape may be played back automatically, since the positioning information for a specific observation is available in the log file.

2.2.3.5. MODUSR5

NEFF 130 amplifier gain is queried by this sub-program. NEFF 130 amplifiers have been used to augment the Precision Filter System 6000 amplifier and filter system by adding additional channels to the test scenario. Although the NEFF 130 amplifiers provide band limiting filters, and adjustable gains, the phase matching, dynamic range, and low noise characteristics are not as acceptable as the Precision Filter System 6000. However, the requirement for additional channels with the availability of gain information to DDAPS is met by module MODUSR5. Since the sub-program is not normally used, the data logging functions (GET MEAN DATA, and LOG

DATA POINT) should be modified (in sub-program MODUSR2) to log the NEFF 130 gains, if this capability is required.

2.2.3.6. MODUSR6

Data packet transfers from DDAPS to the Spectral Data System (SDS) were used during an 8'TPT test in June, 1989. The SDS relied on this data to perform data acquisition and processing of microphone data acquired simultaneously with hot-wire data acquired and processed by DDAPS. This link provided data retrieved from the wind tunnel data acquisition, as well as gain information retrieved from the amplifier filter system. Since the dynamic data of interest to the SDS was stored on additional tracks of the same FM tape that contained fluctuating hot-wire data, start and stop times were made available to SDS as well. The hardware link was via a pair of dedicated GPIB interface cards, with DDAPS acting as system controller. [This HPIB interface should not be confused with the HPIB interface between the wind tunnel data acquisition computer and DDAPS.]

2.2.3.7. MODUSR8

A Digital voltmeter and scanner interface provides accurate voltage measurements from a variety of sources. Both AC (rms) voltages and DC voltages are measured, and the capability to measure resistance is also incorporated. This mechanism has been used to substantiate anemometer measurements, and is also utilized as a reference measurement when calibrating the analog portion of DDAPS.

2.2.3.8. MOD7

The integration of an array processor into DDAPS has boosted performance more than any other modification - including the inclusion of an optical disk for data storage.

Array processor integration parallels ACQUIRE routine MODARITH, since many of the functions are intended to enhance the performance of existing arithmetic functions implemented in MODARITH. It is easy to recognize that many of the functions are actually MODARITH functions, that are now available as a "turbocharged" function. For example, REMOVE MEAN becomes AP REMOVE MEAN.

3. OPERATION

ACQUIRE operation is covered in a separate document - the ACQUIRE System Operating Manual. The operator interfaces should be familiar before proceeding with actual test support. The functions available from Data Laboratories in ACQUIRE are defined in the System Reference Manual, and have been augmented by the functions defined in Appendix A. The operator will gain an understanding of the utilization and operation of DDAPS in the System Description and Theory sections.

Operation of DDAPS begins prior to tunnel operation. Configuration of both the ACQUIRE software and the high speed digitizer is accomplished from within the software. Configuration files are generated by the software and are recallable either at power on time or from within a sequence program.

3.1. System setup - hardware

The initial configuration of hardware is accomplished only once. The assignment of device addresses is as follows:

floppy disks	700,0;700,1	
printer	701	
20 MB disk	1403,0	
67 MB streamer tape	1403,1	
plotter	705	
precision filter	1619	
high speed digitizer		
(control link)	708	GPIB
(data link)	12	GPIO
or (data link)	708	GPIB
(change control variable in MODDL6000 program)		
host computer link	801	
55 MB disk	1402	
340 MB disk	1407	
1600 BPI tape	1601	
650 MB optical disk	1406	
FM tape interface	17	

3.2. System setup - software

The ACQUIRE operating system is configured to operate within the memory constraints of 8 MB, 14 channels of digitizers, and 14 channels of data memory. Refer to the ACQUIRE System Operationing Manual for further details.

The discussion that follows assumes the following file structure is in place:

```
                                : ,1403   Winchester disk
/SYSBAS50  - the boot system

                                : ,1406   Optical disk
/TEST: ,1406
    ACQ           the ACQUIRE program (DDAPS software)
    AUTOSEQ       automatic initializing sequence program
    S_INIT        initializing sequence program
    S_CALIB       calibration sequence program - selects the
                  digitizer calibration sequence
    S_CALTAPE     tape calibration sequence program
    S_RUN         data acquisition sequence program
    S_PLAYBACK    tape digitization sequence program
    S_PROCESS     three wire data reduction sequence program
    S_REDUCE      single wire data acquisition and reduction
                  sequence program
    S_PSD         power spectral density sequence programs
    SWTEST6000    digitizer configuration file
```

/TEST/DATA: ,1406

(No files are required to exist in the DATA sub directory, but the files shown are representative of those that will be generated. Files starting with 345 are the log files, C345_psn is a typical coefficient file, the file starting with R is typical of digitized fluctuating data, and the files starting with V, D, and T are computed velocity, density, and temperature fluctuations.)

```
345          logfile (raw data)
345A         logfile (probe 1 data)
345B         logfile (probe 2 data) if more than 1 probe
345C         logfile (probe 3 data) if more than 2 probes
345D         logfile (probe 4 data) if more than 3 probes
C345_psn     coefficient file for probe serial number psn
RA622501     digitized data according to file naming
              conventions in Table 2.
VA6225_      computed velocity fluctuation data according
              to file naming conventions in Table 2.
```


DA6225___ computed density fluctuation data according
to file naming conventions in Table 2.
TA6225___ computed temperature fluctuation data according
to file naming conventions in Table 2.

/AP: ,1406

APAPPS.S Array Processor executive program.

3.3. ACQUIRE installation

The installation procedure for the ACQUIRE software defined by the manufacturer allows the software to be configured for existing hardware, the available memory in the computer, the number of digitizers in the digitizer chassis, and the maximum number of channels in the computer memory at any one time. From the manufacturer, the ACQUIRE software is configured to support a large data buffer. This buffer may be segmented into several channels of 256K samples each, or more channels if the required samples per channel is reduced. These channel configurations are accomplished with functions MEM START and MEM LENGTH.

The inclusion of application sub-programs has already been accomplished in the version available from the author in a two-disk set (ACQUIRE_1 and ACQUIRE_2.) Some of these SUB programs may be inappropriate for certain tests, and should be deleted (using the HP BASIC command DELSUB) to allow for a simpler program using less memory and taking less time to execute.

3.4. Tailoring the test setup

Sequence programs (see Appendix C.) provide the tailoring mechanism to allow the operator to adjust to changing requirements. These requirements are usually driven by the research, and include data bandwidth, channel count, selection of realtime processing or playback processing, and the availability of prerecorded data on FM tape, and the availability of associated equipment - like a wind tunnel data acquisition system, and pre-calibrated hot wire probes.

The version of ACQUIRE automatically loaded from the optical disk file /TEST/ACQ:,1406 and the sequence programs provided on the optical disk in subdirectory /TEST:,1406¹ are tailored to support the following scenario:

One three wire probe (pre-calibrated) feeds six digitizer channels, channels 1, 2, and 3 for fluctuating data, and channels 4, 5, and 6 for mean data. The test environment requires a fluctuating data bandwidth of 1 Hz to 2 KHz, and the sampling rate is 12,500 samples per second per channel. Over 10 seconds of data (128K samples) are digitized per channel. The mean data is DC to 10 Hz (for anti-aliasing protection) and is sampled at a rate of 400 samples per second per channel. Over 10 seconds of data (4K samples) are digitized. Gain selection of the amplifier/filter subsystem is manually set to provide maximum gains - without overdriving succeeding filters, amplifiers, and digitizer amplitude limits. The digitizer limits are set to 2Vp-p (± 1 V) in the fluctuating channels, and 20Vp-p for the mean channels.

Sequence program S_INIT configures all six channels, both in the memory allocations, and in the display screen format. If the FULL CAL option is selected, S_CALIB, provides real time calibration of the precision filter system and the digitizer. S_RUN takes data. The existing wind tunnel data acquisition system provides a trigger to DDAPS via the GPIB interface, which emulates the ENTER key, allowing an observation to be recorded and processed. Wind tunnel environment data is received from the wind tunnel data acquisition system, and the mean and fluctuating wire voltages are received from three anemometers. The mean data is logged in the logfile, and the fluctuating voltages are stored alongside the logfile on the optical

1. The program and the sequence programs are also available from the author on 3 1/2" floppy disk. A software license is required from Lucas Industries, as is described in the Acknowledgements section of this report.

disk. After each observation, velocity, density, total temperature, massflow, and total pressure fluctuations are calculated, and velocity, density, and total temperature fluctuations are stored on optical disk. The rms values of the fluctuations are added to the logfile. The FM tape is not required.

If this scenario is not appropriate, then the hardware setup and the sequence programs must be modified. When first starting DDAPS, allow the automatic sequence to continue until the operator is asked to select a calibration type - select "Nothing". The sequence program will stop. Using the various functions available in ACQUIRE (starting with SEQUENCE MENU), change the sequence programs. Note that ACQUIRE will not overwrite an existing sequence file. To create a new file of the same name, Acquire must be stopped - press the Shift and Stop keys - and utilize the HP BASIC PURGE command to delete the sequence program. Restart ACQUIRE - press the CONTINUE key - and STORE SEQ PROG. Refer to the ACQUIRE System Operating and Reference Manuals, and Appendix A to select the needed functions.

Sequence programs exist that have supported a variety of other scenarios - uncalibrated wires, multiple probes, mean quantities processed by the wind tunnel data acquisition system, parallel FM tape operation, etc. These other files, although not specifically referenced here, are available from the author, and may provide excellent examples from which to begin. Note, however, that early sequence programs may invoke functions that have been modified since they were first used, and undetermined results may occur.

3.5. Getting started

All equipment containing analog data circuitry - the precision filter, the Multitrap dl6000 digitizer, and the digital voltmeter (if available) - should be turned on at least 1 hour prior to processing analog data.

Thermal stabilization of the adjacent anemometers requires a warmup period of 1 hour.

The adjacent wind tunnel data acquisition computer should be on and available to trigger the data acquisition sequence, and to provide wind tunnel data.

All computer peripherals, the array processor, the digitizer, and the precision filter should be turned on before the cpu. The cpu should be turned on last (all three bottom chassis - top to bottom).

The computer will locate a system to boot, and load the system. When the system has been loaded, begin the initialization of the software environment by typing:

```
LOAD "AUTOST",1 <CR>
```

where <CR> means "press the ENTER key"

This action initiates a sequence of events:

The AUTOST file is loaded and executed, which makes some preliminary file assignments, and then loads the DDAPS program.

The DDAPS program loads, and begins to perform an internal software initialization, which includes a determination of the availability of resources necessary for operation, initialization of data, and initialization of certain hardware - specifically the array processor, if it exists.

When initialization is complete, DDAPS tries to locate, and finds the AUTOSEQ sequence program file.

3.6. Sequence program - initialization

The AUTOSEQ initialization sequence program has been generated, which will determine a sequence of functions to be performed when DDAPS begins to execute the first time after loading at power up time. This sequence program selects and loads another sequence program - S_INIT.

S_INIT, once loaded initializes a variety of control parameters, which define the display and data environment. S_INIT then loads S_CALIB.

S_CALIB assists the operator in the calibration of the digitization process. The operator must select a calibration mechanism by number, and then press ENTER.

When selecting FULL CAL, AC ONLY, or DC only, a calibration of the digitizer is initiated within S_CALIB. This calibration is necessary prior to digitizing real time data. The operator is requested to provide the calibration signal, and that signal is available from the precision filter subsystem calibration card. The calibration card is controlled from the precision filter front panel, and the output from the card is

available at a rear panel connector. The calibration signal must be cabled to all analog inputs to DDAPS (at the patch panel, which connects to the input of the precision filter) and to the digital voltmeter, which acts as the reference measurement unit.

When selecting tape calibration, sequence program S_TAPECAL is loaded and invoked.

S_TAPECAL provides the sequence for calibrating the digitization process (configured for three tracks, or channels) when the calibration reference signal has been previously recorded on FM tape. This signal is usually a 1KHz sine wave at 1Vrms and is recorded on all tracks in use. The calibration information is transformed into internal parameters that will then be automatically applied to digitized data.

See Appendix C for listings of the sequence programs.

3.7. Sequence program - acquisition

The S_RUN run sequence program is activated by the S_CALIB sequence program, which defines a sequence of functions to be performed:

- acquire mean data

- log calibration data

- digitize and store fluctuating data

- print a report displaying many parameters of the current observation whenever the operator presses ENTER. See Appendix C for a listing of the run sequence program.

3.8. File transfer to PC

Several methods exist to transfer files to a Disk Operating System (DOS) environment. Only one is implemented internally to DDAPS, and is described below. Other methods include the use of UNIX utilities to convert the files to UNIX format, then, using network facilities like PC-FTP, transmitting the files to a DOS environment.

3.8.1. LOGFILE TO PC

The probe log data files - one or all - can also be transferred to a PC via a dedicated GPIB cabled between DDAPS and a PC. The PC BASIC program "XFR.HP" (see Appendix D) should be started first, and then, before providing the requested file name, invoke the DDAPS function LOGFILE TO PC. Refer to the relevant function sheet in Appendix B for details on proper configuration prior to starting the transfer. When the file name, which defines where the data is to be stored, is entered into the PC, the transfer will begin.

3.8.2. 3.8.3. File conversion in DOS

More commonly, Oswego software, running in the DOS environment is used to convert the logfiles (in HP BASIC logical interchange format (lif)), to ASCII Disk Operating System (DOS) format files. Refer to the Oswego reference manual for operating instructions. A conversion control file is required to convert binary data format files (BDAT) to ASCII DOS format. This file resides in a DOS environment, and may be called C:\346\CONT.

It contains:

RANDOM

1,1:1s vN vO 100s 1r 20s 20r

P=N*O

2,2:Pr

NOTE: 100 is the number of variables in the logfile. Early logfiles contain only 50 variables, so the conversion control file would have to be modified for those files.

4. SYSTEM THEORY

4.1. HARDWARE

The system is configured to functionally separate areas of signal conditioning, digitization, processing and display in a fashion that maximizes the data throughput. Special consideration was given to aggregate bandwidths of digitized fluctuating data.

4.1.1. Data Flow

The usual source of signals processed by DDAPS is a constant temperature hot wire anemometer. The output of the anemometer is a signal proportional to the heat transferred from the heated wire to its adjacent environment. This voltage consists of a mean quantity (about 2 to 30 volts DC) and a fluctuating quantity (about 1 to 500 mV AC) superimposed on the DC signal.

The acquisition and processing of hot-wire anemometry requires the acquisition of the environment conditions as well as the anemometer signals. However, this system separates signals into mean quantities ($< 1\text{Hz}$) and fluctuating quantities ($\geq 1\text{Hz}$).

4.1.1.1. Mean Quantities

The fundamental mean quantity is the DC voltage received from the anemometer. Mean quantities are also received from the test environment, and represent such quantities as total and static pressure, total temperature, as well as recording parameters such as run, test and point numbers. An existing wind tunnel data acquisition system is often used to digitize all mean quantities and to provide computed environment parameters such as Mach and Reynolds numbers, velocity and density.

Parameters received through a data packet connection from another system are mean quantities, but are processed elsewhere - in the wind tunnel system, for example - and the signal processing, signal resolution, and calibration is the responsibility of that system.

Mean quantities - represented by voltages - can also be processed directly by DDAPS. This technique under-utilizes both the filter subsystem and the high speed digitizers, but if the channels are available, this method is quite serviceable. In this approach, each anemometer is cabled to two channels in DDAPS, and one channel would be processed as fluctuating data (see below) and the other (DC) channel would be DC coupled in the filter system, with the upper bandwidth (the low pass filter) set low enough (e.g. 10 Hz) to act as an anti-aliasing filter for the digitizer. The sampling rate for the DC channels in the digitizer is obviously set much lower than the channels used for fluctuating data. Since the Data Laboratories Multitrap digitizer time base is designed to accommodate two separate sampling clocks, the configuration is easily accomplished via digitizer setup menus provided by ACQUIRE function SET UP SWS.

4.1.1.2. Fluctuating Quantities

The fluctuating quantity is separated from the anemometer signal by the amplifier filter system, where the DC is blocked - first by AC coupling (essentially a series capacitor) at about 0.156 Hz, amplified to about 8 Vp-p, hi pass filtered (usually at 1Hz), amplified again to about 8 Vp-p, low pass filtered (variably at 1KHz, 2KHz, 5KHz, or any other upper band limit), and gained again to achieve about 2Vp-p.

This band limited fluctuating signal is then passed into the high speed digitizer.

4.1.1.2.1. Bandwidth

Bandwidth is the basic measurement of performance of any system. Although the channel bandwidth is most often specified - in this case 0 to 100Khz for 8 channels, and 0 to 250Khz for 4 channels - the entire data path, or "pipe" has a bandwidth at each part of the path. Where the bandwidth is minimum, a "bottleneck" occurs. In the current configuration, the bottleneck is the cpu I/O bus (1 MB/second).

4.1.1.2.2. Aggregate Bandwidth

Aggregate bandwidth is a concern where multiple channels of data are multiplexed into a single path. The bandwidth of all the channels added together form the aggregate bandwidth. If the bandwidth of the path is less than the aggregate bandwidth, a bottleneck exists. If the system is configured for 12 channels at 512K samples per second, then the aggregate bandwidth would be 6144 KB - over 6 MB. Obviously the cpu, with only a 1 MB bandwidth, is a bottleneck.

4.1.1.2.3. Data Buffering

To overcome the bottleneck problem, data buffers exist in the individual digitizer channels, in the cpu (main memory), and in the array processor (data memory). This allows the batch processing of data within a subsystem at rates that would choke the interconnecting paths between subsystems.

4.1.2. FM Tape/Time Code Generator Interface

To accommodate large amounts of data previously recorded, or to provide more channels, and/or longer durations of data at wide bandwidths (to the

limit of the tape recorder) than the digitizer buffers can accumulate before they fill up, a tape interface was necessary to automate the process of recording and playback.

The interface which allows automatic recording and playback of data on FM tape uses a custom digital interface which connects existing equipment into the ACQUIRE software, and allows control of the hardware from within a sequence program either for acquisition or playback of data.

The Honeywell Model 96 tape recorder is an intermediate band recorder with 80KHz bandwidth at 120 inches per second, 40KHz at 60 ips, 20KHz at 30 ips, etc. Each of the 28 tracks has a 528 dB dynamic range.

The time code generator provides an IRIG B time of day signal which is recorded on the tape (usually track 28). This time is also readable in digital form at the interface. The time code generator can also be switched (front panel) to read (translate) the IRIG B time signal previously recorded on the tape.

The HP 98623A BCD interface card, which plugs into the cpu I/O bus, is used in a binary mode. This card was selected because it has both more digital inputs than a GPIO card, and it also has 8 digital outputs to control the tape motion, speed, and mode functions.

Connecting all three of the above components is a custom interface, which provides the physical connections, the electrical compatibility, and minor logical functions in a small chassis. Power for the interface is obtained from the BCD card, where a current limiting resistor in the +5V source was jumpered to provide adequate power. Appendix C provides technical details of the design and implementation of the hardware. Software is implemented in SUB MODUSR4.

4.1.3. NEFF 130 Amplifier Interface

Prior to the purchase of 8 channels of amplifiers and filters, a test was supported that required additional signal conditioning. These additional channels were supported with existing NEFF 130 amplifiers, which were capable of providing a digital binary-coded-decimal (BCD) gain setting status. This gain status was presented to an HP98623A BCD interface card, which was queried by the software. The interface chassis provides the physical connection of 10 NEFF 130 amplifiers to one BCD card. No logic exists in the chassis. The interface chassis is actually a terminal box.

Appendix D provides technical details of the design and implementation of the hardware. Software is implemented in SUB *MODUSR5*.

4.2. SOFTWARE

4.2.1. ACQUIRE

The basic operation of ACQUIRE is not a part of this report. However, some understanding of the premise of its design is helpful in both programming applications and in utilization of the full spectrum of features to be found in the basic package.

ACQUIRE's structure is that of a loop - called *Mainloop* in the code. In each pass through the loop, several basic elements of operation are accomplished: functions queued are executed; a status of the system is performed; operator input devices are queried; sequence program operation is tested, and processed if running; and the display is updated if necessary.

The function is the basic element of operational structure. Functions exist in program modules called SUBs, usually many related functions are grouped together in a single SUB. Each function is capable of performing a variety of actions, based on a variable commonly called *Routine*, where the most common action is 31 - perform action. Other actions are 21 - get existing value from the functions variable, and 22 - update and display the variable.

During initialization - at first load and startup - and before the main loop is entered, each SUB is activated, and told to return the functions (and relevant information about the function) to the main program.

Whenever a request for a function is detected in the main loop, the relevant characteristics - which SUB contains the function, and what input/output support will be required, and which action codes need to be invoked to fully implement the function - are placed in an execution

stack. This stack is then "executed" in *Execcmd*, which invokes *Execitem*, which invokes *Execstack*. *Execstack* actually controls the execution of the function, by calling *Routine*.

SUB *Routine* is the "traffic cop" for invoking a variety of SUBs containing functions. Each SUB is assigned 100 function numbers, and these numbers are coded into SUB *Routine*. If a function is invoked, either by the operator, or programmatically, a "CALL Routine(X,Y)" results - where X is the action desired within the function, and Y is the function number. *Routine* then calls the actual SUB that implements the function, passing along the action code, and the function number.

4.2.1.1. Configuration files

Configuration files used by DDAPS include acquisition setup parameters, hardware configuration parameters, and default display parameters. They are set, saved and stored by a variety of functions. A "standard" set of configuration files is provided only as a beginning point for the new user of DDAPS; these files may not be accurate for a specific test and/or data reduction scenario. These files are listed in Appendix D. Configuration files, unlike sequence programs, are not "edited", but are internally formatted binary files that contain values from internal control variables. The values are stored by functions STORE SYS VARS, STORE COEFS, STORE PLOT, etc.

4.2.1.2. System variables

A file structure is maintained in the ACQUIRE software for containing a wide variety of currently selected operating parameters. This mechanism allows the operator to interactively select preferred operational

conditions, and then store the "sysvars" on disk for later retrieval. These parameters include, but are not limited to, display format, memory length for each channel, waveform file names, waveform channel selection, system variables file name, binary switch name, plot file name, and sequence file name. To recall a specific set of system variables automatically at power on time, the system variables are stored in a file called "AUTOVARS".

4.2.1.3. Binary switches

The high speed digitizer is configured using an interactive session to select sampling rates, gains, trigger modes, data block size, etc., and then the configuration of the binary switches within the digitizer are saved in a binary switch configuration file. To recall a specific configuration for the digitizer automatically at power on time, the binary switches are stored in a file called "AUTOSW".

4.2.1.4. Plot setup

ACQUIRE provides the mechanism for plotting fluctuating data.

The format of a plot is defined interactively and may then be saved on a plot file. The actual data is not saved in the file. Once the waveform channel in memory is selected, the position, scaling, and labeling of the axis is defined, and waveform labeling is determined. Once all channels are positioned and defined, the plot title is defined, and the plot configuration is saved to disk in a format readable by ACQUIRE.

Scalar data plots become possible with enhancements to ACQUIRE, and scalar plots are described in section 4.2.3.3.

4.2.1.5. Sequence program files

The sequence program files provide a mechanism for specifying a series of functions to be accomplished. Generic initialization, calibration, acquisition and processing sequence program files are listed in Appendix C. These files must be tailored for a specific application before they are run.

Once the configuration files and sequence files are configured, the system is a "turnkey" system. Turn on the hardware, allow the hardware and software to be configured according to pre-defined sequence programs, and press a button to initiate the acquisition and processing of hot-wire data.

4.2.2. APPLICATION ENHANCEMENTS

Most enhancements involve new SUBs - MODUSR1, MODUSR2, MODUSR3, MOD7, etc. A few modifications to the baseline ACQUIRE software were made to fix minor bugs, and to allow some expansion of the baseline capabilities.

4.2.2.1. ACQUIRE Expansion

The setting of the system clock has been turned into a distinct SET TIME function, rather than having it be a part of system initialization.

The sequence program mechanisms were enhanced to allow a more complex sequence.

Several sequence language commands have been added:

WAIT !, like WAIT *, stalls the sequence program until a process completes. WAIT * waits for the Multitrap dl6000 high speed

digitizer to complete its current task; WAIT ! waits for all user functions to complete. Since only a few user functions currently return control before completing (FMAUTORECORD, and GET MEAN DATA), only a few user functions could be holding up the sequence program. The purpose is to allow concurrent high speed digitization, mean data acquisition, and FM tape recording. Once these actions are invoked in a sequence program, the WAIT ! and WAIT * language commands stall the acquisition process until all data has been acquired.

SHOW is a renamed PRINT, and PSHOW is a renamed PPRINT. The renaming of these commands simplified the naming of several application functions - PRINT LOGFILE, PRINT TAPE LOG, PRINT LAST OBS, etc.

SHOW "", with a text string containing only blanks, clears the screen. This is useful at the beginning of a sequence (with GRAPH OFF) to assure that parameters and instructions to the operator all appear on the screen at the same time.

Expressions imbedded in braces [...] were made to be allowed anywhere in the sequence line, and MODSEQ was also modified to allow multiple occurrences of imbedded expressions in the same line - even within a quoted string.

Simple arithmetic operators (binary) were added: (+), (-), (*), (/), and (^).

These changes allow sequence program lines like:

```
10 GRAPH OFF
20 SHOW " " clears the screen
100 CONST K=1.0
111 CONST K= 1{+}[RMS(1)](/)[MAX(1)] brackets ... left to right
120 SHOW " Set the calibrator to [CONST K] Volts (rms)"
125 SHOW " and to [MEAN(1)] Volts DC."
130 SHOW "Press ENTER when ready"
200 WAIT ?
```

```
220 SHOW "Collecting data"
230 GET MEAN DATA
240 FM AUTORECORD
250 ARM/TF
250 WAIT !                               wait for mean data and tape recording
260 WAIT *                               wait for digitizer to complete
270 LOG DATA POINT
280 SHOW " "
290 FILENAME
295 STORE DATA
300 SHOW "DATA TAKEN FOR [CURRENT OBS] IS STORED ON DISK"
```

4.2.2.2. APPLICATION Additions

4.2.2.2.1. Hot wire data acquisition

The ACQUIRE software system has been augmented to support the specific requirements of hot wire anemometry systems currently in use at NASA Langley Research Center in the Experimental Methods Branch of the Fluid Mechanics Division. Software design and implementation follows both form and style of the supplied ACQUIRE software, maintaining the appearance of a seamless environment within ACQUIRE. This feature results in a software system for an instrument that has evolved from a waveform recorder to a hot wire anemometry system designed to acquire and process both mean calibration data and fluctuating hot wire data. The end result is an easily used flow diagnostics instrument that minimizes the researchers workload.

With hot wires, there are more often two concurrent tasks: calibration of the wires, and acquisition of the dynamic, or fluctuating data. This system is designed to calibrate and process three wire probe data. The current implementation supports four three wire probes, and acquires, processes, and stores them separately.

Once the instrument is configured, each data observation is acquired with the push of a single button -- "one button recording". If the wind tunnel

computer is capable of triggering the process in DDAPS - as was the case in the 4" pipe flow test, then "no button recording and processing" is achieved. Data processing can require a few more button sequences, but only because the researcher wishes to provide more direction in the data reduction process.

For the 8'TPT flow diagnostics test in January of 1988, the hot wires had not been previously calibrated, so concurrent calibration data and dynamic data acquisition was necessary. Since a full set of calibration data is necessary to generate accurate sensitivities, no post processing could be done until all data points had been acquired. However, the availability of pre-calibrated probes in the 4" pipe flow test, allowed post processing in near-real-time, to produce actual turbulence results at the rate of 20 observations per hour - in an unattended mode.

4.2.2.2.2. Calibration

Calibration of three wire probes requires a complete data system to acquire mean (static) conditions of both the operational environment and of the hot wire values.

It is assumed that hot wire sensitivity is a function of velocity, density, and total temperature.¹ To determine the sensitivity to each variable, the mean voltage output of each hot wire must be measured at each combination of velocity, density, and temperature. The tunnel run schedule is configured to assure that adequate data points are taken to provide a realistic profile of sensitivities.

The run schedule is also selected to expose the hot wire probe to the highest dynamic pressures first, so that if a wire is going to break, then the least amount of tunnel time will be lost.

Each of the three hot wires are operated at different overheats, to achieve a wide separation in sensitivity between each hot wire.¹

The calibration data consists of mean values, which do not require the high sampling rates normally invoked to digitize the dynamic data, so the calibration data is not usually acquired through the high speed digitizers. An existing wind tunnel data acquisition system can be used to collect the data, which then transfers it through a GPIB link to the DDAPS computer. Data packets of data may be averaged.

4.2.2.2.3. Log files

This calibration data is stored in a log file (also called an observation file). Other related mean data is also stored in the observation file:

- tunnel conditions,
- test identification parameters,
- amplifier gains,
- FM tape start and stop times,
- auxiliary data such as RMS microphone readings, and
 - amplifier gain settings, and
- simple calculated data such as density, velocity,
 - static temperature, the logs
 - of a variety of data, and the
 - products of the logs of a
 - variety of data.

The log file is a collection of files, not a single file. The primary log file is a raw data file, where collected data, and relevant test environment data is first logged. Its file name is the name defined by function LOG FILENAME. Other files, one for each probe, are also generated, and certain data is copied from the raw file to the probe file. Other data relevant to each probe is computed, and placed in the probe

file. These files contain data prepared in subroutines *P1_vars*, *P2_vars*, *P3_vars*, and *P4_vars*. These files have the log filename appended with "A", "B", "C", and "D" respectively.

The formats of these files conform to the internal file format of the HP 98820A Statistical Library.

The internal format of the observation files is carefully selected to conform to the format specifications of an existing statistics package. The 98820A Statistical Library from HP has historically been the statistics package used to reduce the hot wire calibration data, so the file format was made compatible with that package. (A minor deviation: since the subfile concept of the Statistical Library is not required, the first two values of the subfile array in the header are used to hold the last observation number which has logged data, and the test number of the current test.) The number of variables allowed (50) in the statistics package was found to be insufficient, and was changed to 100 in ACQUIRE and in the statistics software. The structure of the statistics program was also combined into one file, which enhanced execution performance. The modified Statistical Library is stored on hard disk, but since the functionality has not changed, the modifications are not otherwise documented.

4.2.2.2.3. Coefficient Files

The coefficient file is a binary data (BDAT) file containing 30 real coefficients in a 3 X 10 array. Up to 10 coefficients (8 are used) per wire are entered and stored. Function COMPUTE SENS utilizes the currently loaded coefficients to compute sensitivities for three wires of the current probe (see function CURRENT PROBE).

Only one set of coefficients (one three wire probe) can be loaded from disk into ACQUIRE at a time.

4.2.2.2.4. Dynamic data

The acquisition of the dynamic hot wire data is entirely accomplished by the off-the-shelf ACQUIRE software. ACQUIRE has all the mechanisms necessary to configure the actual data acquisition hardware and the capability to manage the data, once it has been digitized and buffered by the high speed digitizer hardware, which includes the transfer from the buffer to computer memory, and the transfer of the data to disk. These functional modules are "strung together" in a sequence program mechanism, which is also an inherent part of ACQUIRE.

4.2.2.2.5. Fluctuating data

Fluctuating data is either digitized data from the hot wire anemometers, or it is calculated data from the process of computing velocity, density, and temperature fluctuations, which is discussed later. In either case, the result is a time varying data array of samples.

The sheer volume of fluctuating data is worthy of note: since each channel can handle 256K samples (512K bytes) at a time, and there are 3 channels per probe, and mean and fluctuating measurements are taken results in $256K \times 2 \times 3 \times 2 = 3072K$ bytes per observation. For 117 observations (8' Transonic Pressure Tunnel Test 934), 360M bytes of data storage becomes necessary per probe. If 4 probes were used, 1440M bytes (1.44Gb) of storage would be required.

DDAPS collects and generates multiple channels of fluctuating data files for each test condition or observation. These files are related to a

specific record in an observation file, which contains non-fluctuating scalar data related to the observation. The relationship of the fluctuating data file names to the test condition and to the observation file and record within the observation file is specifically defined by convention. The use of a naming convention allows data reduction programs to associate all data files necessary for data reduction and for naming resultant fluctuating data files. Table 2 details both the preliminary naming formats and the final naming formats.

The dynamic data samples digitized by the MULTITRAP digitizer hardware are stored by the waveform recorder function of ACQUIRE on a channel-per-file basis. Fluctuating data names are generated whenever a hot wire calibration observation is logged, so that a naming convention is followed - should a request to digitize hot wire fluctuating data be processed.

4.2.2.2.6. Coefficient calculations

The process of providing the sensitivities necessary to convert three hot wire data arrays into velocity, density and temperature fluctuation arrays first requires that the calibration data be processed to produce a set of coefficients for each of the three hot wires. Since the relationship of the performance of the hot wire is highly nonlinear in relationship to the velocity, density, and temperature, up to 10 coefficients are accommodated (Eight are in use, as shown¹:

$$\begin{aligned}\log E = & A_1 + A_2 \log u + A_3 \log \rho + A_4 \log_0 T_0 \\ & + A_5 \log u \log \rho + A_6 \log u \log T_0 \\ & + A_7 \log \rho \log T_0 \\ & + A_8 \log u \log \rho \log T_0\end{aligned}\quad (1)$$

Since velocity, density, and temperature are all known for each observation (as collected by the DDAPS from the tunnel data acquisition system - in the form of P_T , P_S , and T_T), the most direct solution is through multiple linear regression.

Whenever requested by the operator, a Multiple Linear Regression routine (which is a specifically modified version of the Hewlett Packard routine MLR which was purchased as part of a statistics package) is invoked, which calculates coefficients for each hot wire on each probe. These coefficients can then be stored in a coefficient disk file related to each probe. (This internal MLR routine is not currently implemented.)

Alternatively, coefficients calculated in a separate multiple linear regression package - typically the HP98820A Statistical Library - may be

read from a disk file generated by that package, or, as in the case of the Statistical Library, the coefficients may be manually entered through the keyboard.

4.2.2.2.7. Sensitivity calculations

The coefficients, which represent the hot wire relationship to velocity, density, and temperature - based on the function defined in eq. (1), are combined with specific test conditions, which have been stored in an observation record of the observation file.

$$\begin{aligned} S_u &= A_2 + A_5 \log \rho + A_6 \log T_0 \\ &+ A_8 \log \rho \log T_0 \end{aligned} \quad (2a)$$

$$\begin{aligned} S_\rho &= A_3 + A_5 \log u + A_7 \log T_0 \\ &+ A_8 \log u \log T_0 \end{aligned} \quad (2b)$$

$$\begin{aligned} S_{T_0} &= A_4 + A_6 \log u + A_7 \log \rho \\ &+ A_8 \log u \log \rho \end{aligned} \quad (2c)$$

Functions programmed in MODUSR2 allow the appropriate calibration file to be specified, and the beginning and ending observations and beginning and ending probes to be selected for the computations. For each observation and each probe, the log values of velocity, density and temperature are retrieved from the appropriate record in the observation file. Once the computation is completed for each probe, the resultant sensitivities are inserted into existing, but as yet unused variables in the previously recorded observation.

4.2.2.2.8. Calculating fluctuations

Once the sensitivities have been calculated, the operator may request that the dynamic data for a given set of observations and probes be processed in a way that yields dynamic waveforms representing fluctuating velocity, density and temperature (instead of 3 fluctuating voltages) and with turbulence figures and other statistical performance characteristics.

The equations that define the relationship of voltages to turbulence parameters are:

$$\left[\frac{e'}{E} \frac{1}{G_w} \right]_1 = S_{u_1} \frac{u'}{U} + S_{\rho_1} \frac{\rho'}{\rho} + S_{T_{0_1}} \frac{T_{0'}}{T_0} \quad (3a)$$

$$\left[\frac{e'}{E} \frac{1}{G_w} \right]_2 = S_{u_2} \frac{u'}{U} + S_{\rho_2} \frac{\rho'}{\rho} + S_{T_{0_2}} \frac{T_{0'}}{T_0} \quad (3b)$$

$$\left[\frac{e'}{E} \frac{1}{G_w} \right]_3 = S_{u_3} \frac{u'}{U} + S_{\rho_3} \frac{\rho'}{\rho} + S_{T_{0_3}} \frac{T_{0'}}{T_0} \quad (3c)$$

To solve for the three unknowns $(\frac{u'}{U}, \frac{\rho'}{\rho}, \text{ and } \frac{T_{0'}}{T_0})$ the simultaneous equations are rearranged for matrix operations:

$$\begin{bmatrix} \left[\frac{e'}{E} \frac{1}{G_w} \right]_1 \\ \left[\frac{e'}{E} \frac{1}{G_w} \right]_2 \\ \left[\frac{e'}{E} \frac{1}{G_w} \right]_3 \end{bmatrix} = \begin{bmatrix} S_{u_1} & S_{\rho_1} & S_{T_{0_1}} \\ S_{u_2} & S_{\rho_2} & S_{T_{0_2}} \\ S_{u_3} & S_{\rho_3} & S_{T_{0_3}} \end{bmatrix} \times \begin{bmatrix} \frac{u'}{U} \\ \frac{\rho'}{\rho} \\ \frac{T_{0'}}{T_0} \end{bmatrix} \quad (4)$$

By rearranging again, which involves inverting the sensitivity matrix, solve for the three unknowns:

$$\begin{bmatrix} \left[\frac{u'}{U} \right] \\ \left[\frac{\rho'}{\rho} \right] \\ \left[\frac{T_0'}{T_0} \right] \end{bmatrix} = \begin{bmatrix} S_{u_1} & S_{\rho_1} & S_{T_0_1} \\ S_{u_2} & S_{\rho_2} & S_{T_0_2} \\ S_{u_3} & S_{\rho_3} & S_{T_0_3} \end{bmatrix}^{-1} \times \begin{bmatrix} \left[\frac{e'}{E} \frac{1}{G_w} \right]_1 \\ \left[\frac{e'}{E} \frac{1}{G_w} \right]_2 \\ \left[\frac{e'}{E} \frac{1}{G_w} \right]_3 \end{bmatrix} \quad (5)$$

The computation of the instantaneous velocity, density, and temperature is accomplished as shown:

- ⊙ for each probe and each observation to be processed
 - ⊙ for each of the three hot wires
 - ⊙ the mean hot wire voltage (E) is retrieved
 - ⊙ the gain (G) for the fluctuating hot wire voltage is retrieved
 - ⊙ the three sensitivities (u, ρ, T₀) are retrieved and placed in the sensitivity matrix
 - ⊙ the dynamic data file is retrieved from disk, and placed in memory
 - ⊙ the sensitivity matrix is inverted
 - ⊙ for each of the instantaneous samples
 - ⊙ for each of the three hot wires
 - ⊙ compute:

$$\frac{e'}{E} \frac{1}{G_w}$$

- ⊙ and store in the independent variable matrix

⊙ matrix multiply the inverted sensitivity array by the independent variable array, and place the instantaneous turbulence ratios ($\frac{u'}{U}$, $\frac{\rho'}{\rho}$, and $\frac{T_0'}{T_0}$) in memory

⊙ compute the RMS values of $\frac{u'}{U}$, $\frac{\rho'}{\rho}$, and $\frac{T_0'}{T_0}$

⊙ store the RMS values of velocity, density, and temperature

$\left[\frac{\tilde{u}'}{U}, \frac{\tilde{\rho}'}{\rho}, \frac{\tilde{T}_0'}{T_0} \right]$ for each observation into existing variables in the previously recorded observation

⊙ store the fluctuating velocity, density, and temperature waveforms in disk files for later spectral investigations, using existing functions of ACQUIRE.

4.2.2.2.9. Precision filter system control

The computer control of the filters and amplifiers allows adaptive processing of various hot wire signals, which are dependent on a variety of operational parameters. The software interface allows full control of each functional module within the Precision Filter system - including the calibration module, and allows the interrogation of all status and condition data - including the calibration module. The modules are connected to provide a full calibration sequence, and full operator control, semiautomatic or automatic operation. Calibration sequence program S_CALIB automatically sets up the channels as they will be used in data acquisition, and asks the operator to configure the calibrator for selected calibration signals.

The amplifiers and filters are grouped into hot-wire channels, three to a group (actually, due to a concern for phase matching, one group has 2 channels, and one group has 1 channel). This grouping allows front panel operation to change operational parameters for an entire three wire probe at a time. Operator lockout (functions PF REMOTE and PF LOCAL) inhibits changes while data is being acquired. Status queries allow the data acquisition process access to current gain information on a per-channel basis. Hardware-software configuration compatibility is maintained in support routine PF_INIT.

The Precision Filter System 6000 amplifier and filter system consists of a wide variety of optional modules. The amplifier and filter modules selected for DDAPS are configured to provide a wide gain selection, and both high and low pass filters. The input coupling of the modules are selectable (AC or DC).

To facilitate "per probe" selection which assists the operator in assuring that all three wires are being conditioned consistently, ACQUIRE groups the modules into sets of three (Precision Filters, Inc. calls these modules "channels", and this designation appears both in the Precision Filter manuals, and on the front panel display). Because the first 8 channels (not modules) are identical, two groups of three channels, and one group of two channels are configured. Groups 1, 2 and 3 are each split into "A" and "B", since two different module types (high pass and low pass filters) comprise the signal path. Groups 1a, 1b, 2a, 2b, 3a, and 3b support the first 8 channels. A second set of 4 channels exist, which extends the 100KHz band limit to 250KHz, and uses 135dB filters instead of 85dB filters. These channels use 4 separate module types: pre-amp, high pass filter, low pass filter, and post-amp. Groups 4a, 4b, 4c, 4d, 5a, 5b, 5c, and 5d provide a three channel group and a one channel group.

<u>GROUP</u>	<u>CHANNEL(s)</u>
1	1, 2, 3
2	4, 5, 6
3	7, 8
4	9, 10, 11
5	12

Program MODUSR3 implements the precision filter mechanisms.

4.2.2.2.10. FM Tape Control

The tape control mechanism relies on simple control commands to the tape transport, and on the ability to read the time code generator (TCG). Because the times generated by the TCG are recorded on the tape, and are readable by the program during the recording process, the start and stop times of each observation are logged in the log file. During playback of a selected observation, the start time is retrieved from the log file, and a search algorithm shuttles the tape forward and backwards as necessary to position the tape. Tape position is sensed by the program by reading the TCG (time code reader in this mode) from the tape.

Function FM POSITION provides a mechanism that defines the range of time relevant to the search algorithm. These time limits are the start time of the initial observation (INITIAL OBS) and the stop time of the ending observation (ENDING OBS). FM POSITION actually positions the tape at the first previously recorded observation.

Function FM AUTORECORD records an observation with a duration controlled by function FM RECORD TIME. The start and stop times are noted, and when LOG DATA POINT is invoked, the start and stop times are logged.

FM AUTORECORD when invoked in a sequence program, does not wait for completion prior returning control to the sequence stepping mechanism. Concurrent operations, such as GET MEAN DATA, and ARM can all execute

simultaneously. FM AUTORECORD and GET MEAN DATA can be made to stall the sequence program if a WAIT ! function is inserted in the sequence program, and any application function is in progress. This stalling mechanism parallels the digitizer mechanism where the digitization processes are also no-wait. The WAIT * function stalls a sequence program if the recorder is busy.

Function FM AUTOPLAYBACK plays back an observation beginning at the start time previously logged for the observation.

Tape control relies on simple control of the tape transport - forward reverse stop, record, etc., and the ability to read the time code generator-translator (TCG) time of day information.

The tape motion activity is activated by a 1 bit command line, sent to the interface via the BCD I/O card data out lines. For each command line a separate bit is used. RECORD, for example, would require that both the FORWARD bit, and the RECORD bit be set.

When the front panel switch of the TCG is in "generate", the TCG time is logged at the beginning and ending of a tape recording. When the TCG is in the "translate" mode, a search can be managed by reading the tape time of day when the tape is placed in motion. If the time being read is in excess of the desired time, then the tape motion command is changed to reverse, and the TCG is informed that the tape is moving backward.

FM POSITION retrieves the desired position (actually the desired time) from the initial observation (INITIAL OBS) start time variable in the log file. The tape is then stopped just before that time on the tape.

FM AUTOPLAYBACK positions the tape according to the current observation (CURRENT OBS), but leaves the tape running. In a sequence program, the next logical function might be one involved in digitizing data.

Absolute time limits are checked to assure FM AUTOPLAYBACK does not exceed the search area. These limits are: the initial position start time, and the ending observation stop time.

4.2.2.2.11. Scanner/Voltmeter Control

All functions related to the scanner/voltmeter command the hardware - via GPIB - to perform the desired function, and then, in the case of the voltmeter, a measurement is transferred to the computer, and made available to DDAPS. This area of the program is somewhat variable, and the availability of the scanner, or the type of equipment may vary, requiring some modification as the hardware varies.

4.2.2.2.12. Data Packet Transmission to SDS

The Spectral Data System (SDS) can receive data from DDAPS (see 2.2.3.6., MODUSR6). When the data acquisition sequence begins (GET MEAN DATA), the related functions for queueing SDS to begin processing the observation is invoked. When DDAPS completes the observation, SDS receives a data packet.

4.2.2.2.13. NEFF 130 Amplifier Gain Query

The NEFF 130 amplifiers provide a 4-bit BCD character code defining the gain selected on the front panel of the amplifier module. Each of 10 BCD characters is read in, converted to the actual gain, and put in an

internal buffer, which can be used to update the logfile in the part of ACQUIRE associated with function LOG DATA POINT.

4.2.2.2.14. Data Packets

The data packet concept is one of convention. The two computers involved in a packet transfer agree who is in charge of initiating the transfer (the active controller), and the other computer simply waits for the read or write to be completed by the controlling computer. The passive computer is DDAPS. The first wind tunnel connection (a MODCOMP computer) required that it be in control. The convention remains.

Data packet format is defined in tables 2. and 3. The first and last bytes (STX, ETX) of the data packet are unnecessary for GPIB protocol, and are ignored in the DDAPS implementation. They exist for other implementations, particularly an RS232 implementation sometimes used by the MODCOMP wind tunnel computer (host) when connecting to computers other than DDAPS.

As DDAPS has been used with different wind tunnel computers, the packet messages have been augmented as necessary. The actual data transmitted often changes as the test requirements - and wind tunnel - changes.

In the most recent version of the packet implementation, an "OK" message can be transmitted to the host from DDAPS, informing the host that DDAPS is ready (the READY function). An interrupt subroutine, SUB *Intr8*, is invoked when an GPIB "TRIGGER" is received from the host. This SUB then simulates pressing the ENTER keyboard key. In a sequence program (see the S_REDUCE in C.), the READY function is followed by the sequence command WAIT ?, which stalls the sequence until an ENTER key is pressed. This

mechanism allows the host to control the cycling of DDAPS: the host tells DDAPS to take data.

4.2.3. Utility Functions

Other utility functions are implemented to enhance operational characteristics of the system. The ability to eject plots, and the ability to plot "special" hot wire calibration data, are "plot utilities". The ability to list categories of files on the disk, to purge extraneous files - or groups of files, the ability to copy or move files - or groups of files - to another disk (or tape) are "file utilities".

Function LOGFILE TO PC may be used to transfer observation files containing logged and computed data to another system. A GPIB bus connects DDAPS to the PC, where a GPIB card (National Instruments or HP) is installed. Appendix D contains the PC BASIC program used with the HP GPIB card to receive and store the data on the PC disk.

4.2.3.1. File utilities

The CAT, PURGE, COPY and MOVE utilities are unique in that they allow a selected group of files to be involved. The target and destination disk drive and directory are selectable, as well as the group of files to be involved. The selection criteria is simple (it uses the HP BASIC "SELECT" keyword of the "CAT" command). Any file beginning with the characters specified in the GROUP, will be selected. No wildcards are allowed.

4.2.3.2. Scalar plot utilities

The scalar plot utilities are a collection of hard-coded plots representing data already existing in the log file. The extent of the

number of functions in this category varies as the need arises, and continue to exist as functions for as long as they are useful.

4.2.3.3. Process utilities

The most universal functions in this category are SOUND GOT IT and SOUND ALL DONE. These sounds have become recognizable as the beginning and ending of a sequence of events, and help to keep the operator apprised of the status of the process unfolding within the sequence program. SEQ REMOTE GO has been used to inform a secondary computer system - the Spectral Data System (SDS) - to begin its process. DIGITIZE ENABLE can set or reset a flag, indicating whether or not digitization is required. The value of the flag can be tested programmatically, or DIGITIZE ENABLE can return the value which is useful in a sequence program to control the flow of the program:

```
.  
.   
.   
165 DIGITIZE ENABLE = 1  
.   
.   
.   
230 WAIT ?  
245 LET E=[DIGITIZE ENABLE]  
250 IF E=0 THEN GOTO 300  
260 ARM/TF  
270 WAIT *  
280 STORE DATA  
290 SHOW "DATA TAKEN...."  
300 SOUND GOT IT  
310 GOTO 230  
.   
.   
. 
```

5. SOFTWARE SUPPORT (PROGRAM DEVELOPMENT)

5.1. Array processor program development

The array processor requires that an executive program be loaded that is capable of performing the actions requested by the host software.

Development of the APAPPS.S executable file was half of the array processor integration into the existing ACQUIRE software. The other half occurred in ACQUIRE, invoking a variety of functions available within the array processor.

During the development phase, two processes are delineated in the array processor reference manuals: 1.) generate the BASIC SUBs which invoke the required functions; 2.) generate the executable file APAPPS.S.

When generating the BASIC SUBs, Analogic provided a utility called *BUILD_K_FUNCT*. This utility adequately produced the BASIC code to invoke the array processor functions (called K functions by Analogic). However, the HP BASIC limitation of a 16 bit integer (± 32767) inhibited ACQUIRE from addressing more than 32K samples in the array processor.

BUILD_K_FUNCT was modified to pass the usually defined integer parameters as real numbers. The modified utility is named *BKF*. This change becomes significant, when referencing the array processor reference manuals, since the passing parameters are now real, not integer.

Construction of the executable file APAPPS.S involves another supplied utility. This unchanged utility was used according to the reference manuals.

The ASCII2BAS and BAS2ASCII utilities allow the editing of ASCII files.

5.2. UNIX Operating System

HP-UX 6.5 is available for general use. ACQUIRE is not supported by this operating system, but UNIX can be used for file transfer, network support, file translation, data analysis, etc. C, Fortran, Pascal, and BASIC (RMB-UX) are supported languages. ARPA and NFS networking services are supported. X windows are supported. Many other support services also exist.

The availability of BASIC (RMB-UX) suggests that ACQUIRE could be supported under UNIX. CSUBs - compiled subroutines - would have to be recompiled (Pascal source) before ACQUIRE could run under UNIX. The BCD interface card (used to connect the FM tape subsystem, and to read NEFF 130 amplifier gains) is not supported under HP-UX. The GPIO interface, used for fast transfers of high speed digitizer data, may not be fully supported. The operator interface - screen, keyboard, etc. - would probably require a rewrite to integrate into an X windows environment.

6. SYSTEM SUMMARY

6.1. SOFTWARE

The acquisition of dynamic data, and the storing of dynamic data is a very significant strength of ACQUIRE. But most importantly, the internal design allows application routines to be written into ACQUIRE, which produces a set of software that appears to the user to be a single entity, without seams, and fully integrated.

ACQUIRE, including the application software, and in combination with the hardware is a very versatile dynamic data system, as well as an accomplished data logger.

- ⊙ It controls all the hardware associated with the system.
- ⊙ It manages the hardware and software configuration - via files.
- ⊙ It manages process, or "sequence" files.
- ⊙ It manages dynamic data files and internal arrays of data.
- ⊙ It provides a choice of operator dialogue techniques, including cursor, menu, and command line entry.
- ⊙ It provides data display management.
- ⊙ It provides the waveform plotting capabilities.
- ⊙ The internal log file format allows direct access by a commercially available statistics package.
- ⊙ An array processor provides 90X performance enhancements in the processing of fluctuating data.
- ⊙ A software/hardware link is used with the data acquisition computer to receive mean data values.
- ⊙ A Digital Signal Processing package is included which provides:

Fast Fourier Transforms

filters

power spectrum

transfer functions

7. Planned Future Enhancements

DDAPS needs to be modified in several areas to improve productivity:

Reduce the bottleneck at the output of the digitizers by connecting the digitizers directly to the array processor. This connection would utilize the GPIO output of the Multitrap dl600 high speed digitizer. A cable would connect the digitizer to one of the existing axillary I/O ports of the AP509 array processor. The driving software, SUB Moddl6000 would be most affected by the change. However, the entire ensemble of processing functions would be affected by the change, since the data would not reside in the computer memory (mems), at least initially, and manipulation of fluctuating data would therefore naturally utilize the array processor functions implemented in SUB Mod7, at least until the data was transferred to mems for I/O, display, and plotting by existing functions.

Enhance the DSP performance by implementing the DSP functions in the array processor. At the time of this report, this enhancement is under development.

To provide a better software environment for program development, data reduction and electronic connectivity to the outside world, conversion to the UNIX environment is anticipated. Acquisition of available utility software, such as new statistical software and X Window support would provide a more universal and familiar environment. The conversion to Unix is not trivial, and may require more than running BASIC under Unix, since compiled CSUBs (in Pascal) have to be

converted to the Unix environment. Converting ACQUIRE to a UNIX-compatible language would enhance the support effort, by providing a unified data and program base.

7.1. Connect the digitizer to the array processor

To enhance the transfer speed of digitized data out of the digitizer, a cable would be manufactured to connect the GPIO port of the Multitrap digitizer - currently connected to a GPIO interface card in the cpu - to one of the two auxiliary I/O ports of the array processor. The software modifications would occur in ACQUIRE module *Modd16000*, and would permanently incorporate the array processor into ACQUIRE. This connection would put the data where it needs to be, bypassing the current bottleneck in the 1 MB I/O bus of the cpu. Transfers are anticipated to quadruple in speed, and computations in the array processor could be made to be concurrent with transfers. Refer to Figure 7.

7.2. DSP - Array Processor integration

Array processor integration parallels MODDSP, since many of the digital signal processing functions are implemented in the array processor to enhance performance. The AP-DSP integration has been inserted in the original MODDSP module, and DSP functions implemented in the AP are executed by the AP if the AP was successfully initialized in module MOD7. Status variable Apok is not equal to zero when the AP is available.

SUB Moddsp is being restructured to perform the functions presently executed both in Moddsp, and in the compiled support CSUB, Dspcsub. The implementation is envisioned to utilize existing array processor functions available in Mod7, and like Mod7, to fully utilize the library of DSP function calls provided by the array processor manufacturer, Analogic,

Inc. Logic in the modified Moddsp will sense the availability of the array processor (a variable - Apok - is set by Mod7 upon initializing the array processor) and will then either utilize the array processor or the existing Dspcsub implementation. The array processor promises to be 90 times faster.

7.3. Utilize UNIX operating system

Provide the multitasking, multiuser environment necessary to support concurrent operations, Ethernet (TCP/IP) communications, a choice of programming languages - including interpretive and compiled, and a wider marketplace for software and hardware solutions like nine track magnetic tape support, laser printer support, graphics and statistics support, and data management support.

7.4. Translate ACQUIRE code to Unix compatible languages

HP BASIC 5.0 is the principal language of ACQUIRE. The availability of BASIC (RMB-UX) suggests that ACQUIRE could be supported under UNIX. CSUBs - compiled subroutines - would have to be recompiled (Pascal source) before ACQUIRE could run under UNIX. The BCD interface card, used to connect the FM tape subsystem, and to read NEFF 130 amplifier gains, is not supported under HP-UX 7.0. The GPIO interface, used for fast transfers of high speed digitizer data, may not be fully supported. The operator interface - screen, keyboard, etc. - would probable require a rewrite to integrate into an X windows environment.

Acknowledgements

This work was accomplished for the National Aeronautics and Space Administration under contract NAS1 - 18585 (Task 42) by Vigyan, Inc., Hampton, VA 23666

The ACQUIRE software is owned by Data Laboratories Ltd. - now Lucas Industries. A right to use and modify license has been granted to NASA, and delineates the rights and limitations of use assigned by Data Laboratories to NASA. In addition, a hardware "key" is provided by Data Laboratories, which must be attached to the GPIB (select code 7) to allow execution of the software. The license and key are available in the United States from Lucas Industrial Instruments, Severna Park, MD. (301) 544-8773.

Certain subroutines (scalar plot support, and FM tape control) were jointly developed by the author and Mr. Greg Chichester, Unisys, Hampton, VA.

References:

1. P. C. Stainback, C. B. Johnson, et al; Preliminary Measurements of Velocity, Density and Total Temperature Fluctuations in Compressible Subsonic Flow; AIAA-83-0384
2. P. C. Stainback; Some Influences of Approximate Values for Velocity, Density and Total Temperature Sensitivities on Hot Wire Anemometer Results; AIAA-86-0506
3. Bobbitt, Percy J.: Instrumentation Advances for Transonic Testing. Presented at the Transonic Symposium, NASA Langley Research Center, Hampton, Virginia, April 19-21, 1988. NASA CP-3020.
4. G. S. Jones, P. C. Stainback; A New Look At Wind Tunnel Flow Quality for Transonic Flows; SAE-88-1452
5. Baize, Daniel G.; An Evaluation of the Normal 3-wire Anemometer in a Turbulent Pipe Flow; Masters Thesis, George Washington University, August, 1990
6. ACQUIRE 1.2, issue 2, modification 0, March 23, 1988; System Operating Manual OM0022 System Reference Manual OM0023 Issue A (September 1986, with Addendum February 1988); Data Laboratories Limited. Unites States contact is Lucas Industrial Instruments, 760 Ritchie Highway, #N6, Severna Park, MD 21146; Tel (301) 544-8773

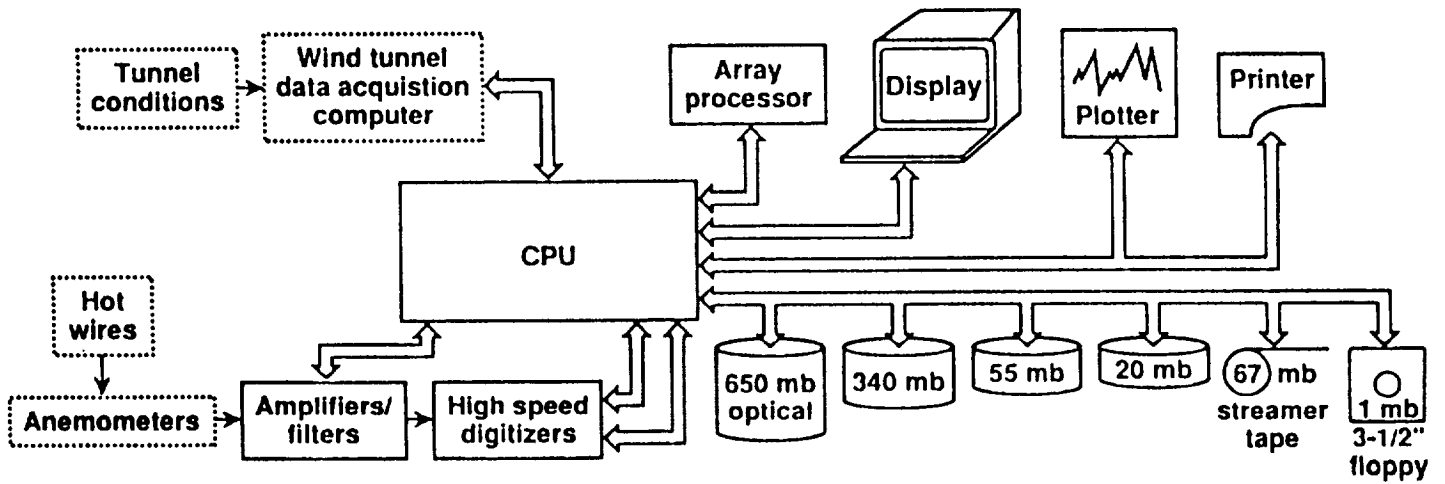


Figure 1. System Block Diagram

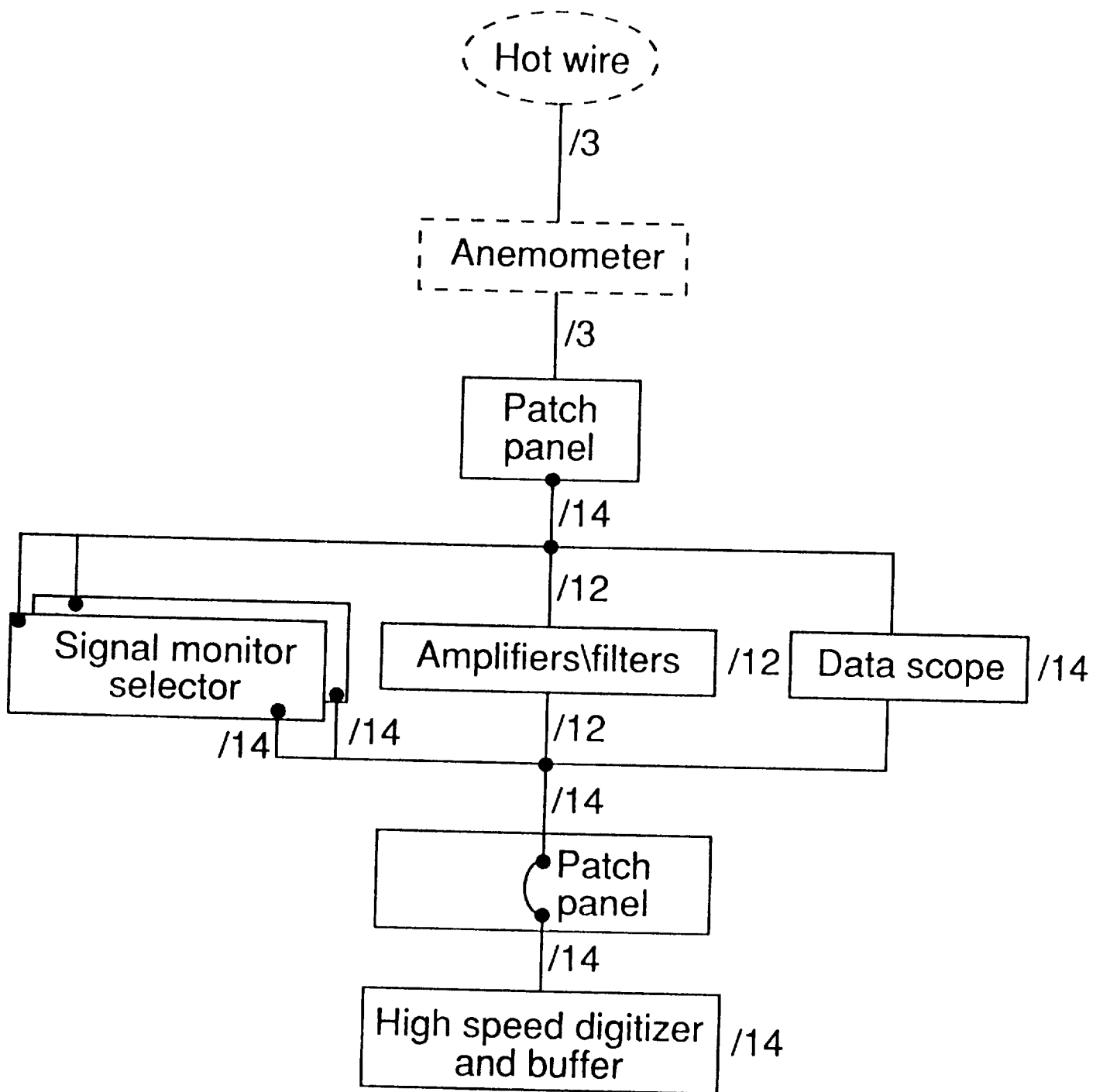


Figure 2. Analog Cabling Diagram - data acquisition

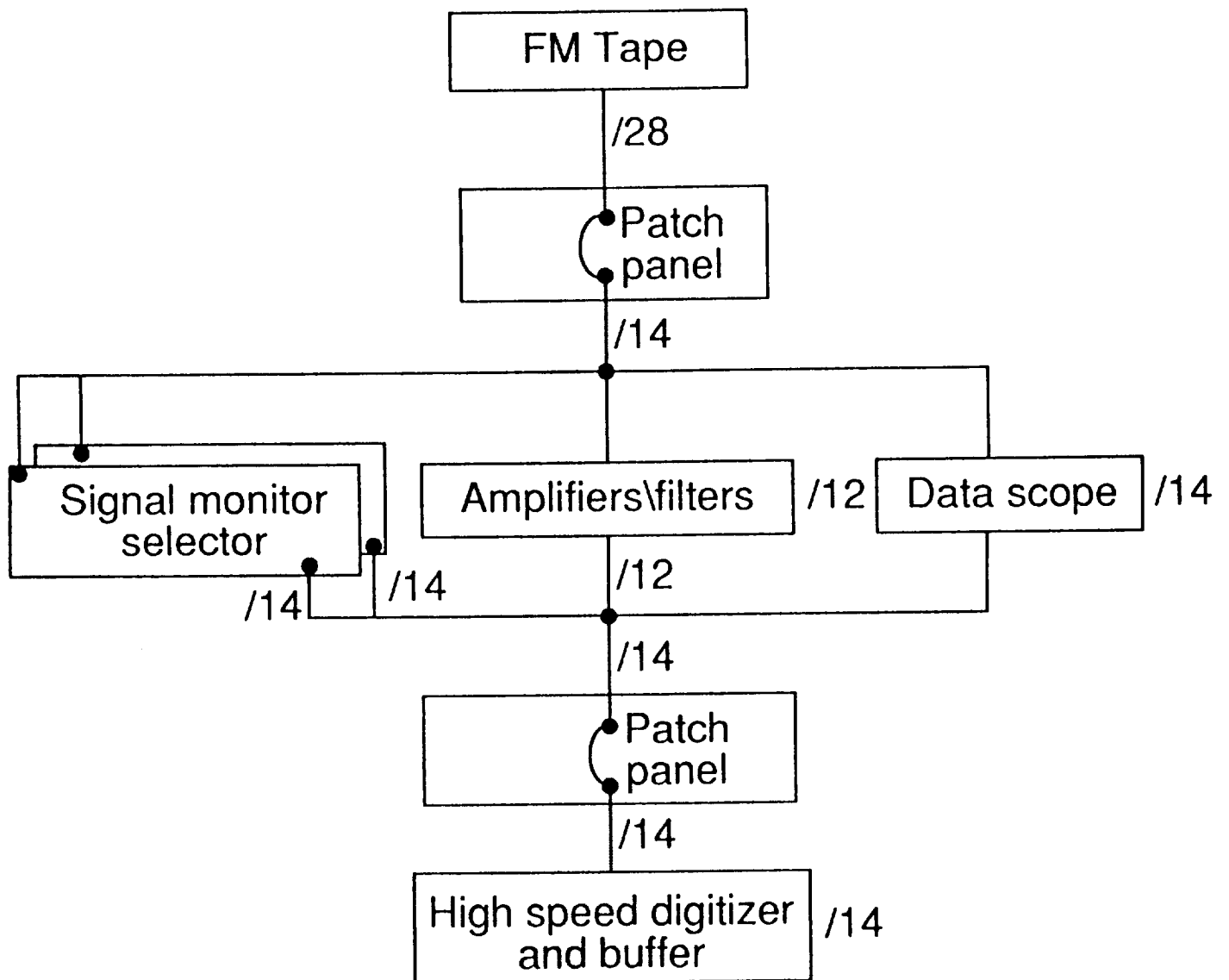


Figure 3. Analog Cabling Diagram - tape playback

8ft TPT HOTWIRE CALIBRATION 9 Dec 1987

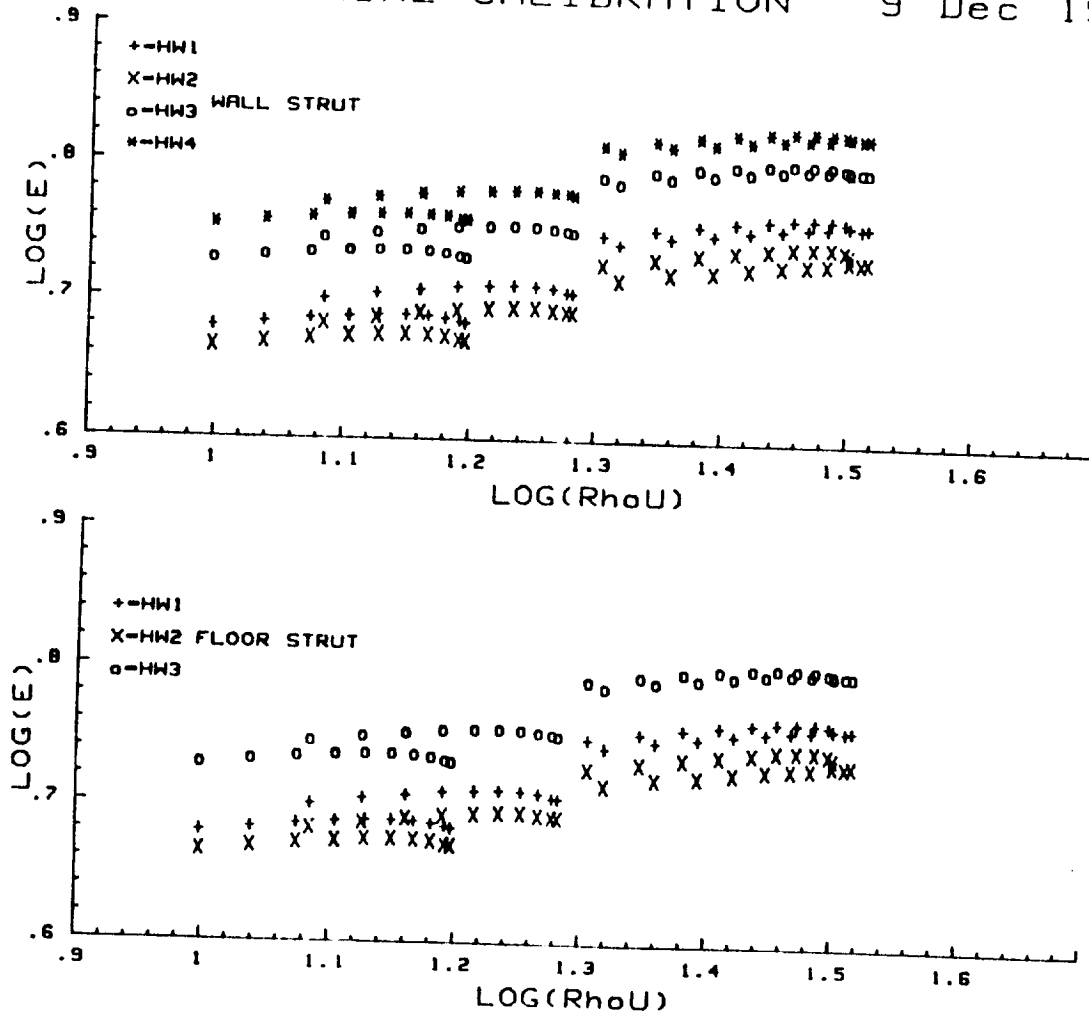


Figure 4. Plot: hot wire voltage vs. mass flow

8FT TEST 934 - DATA REDUCTION -

OBSERVATION # 11

TEST 934
RUN 17
POINT 8

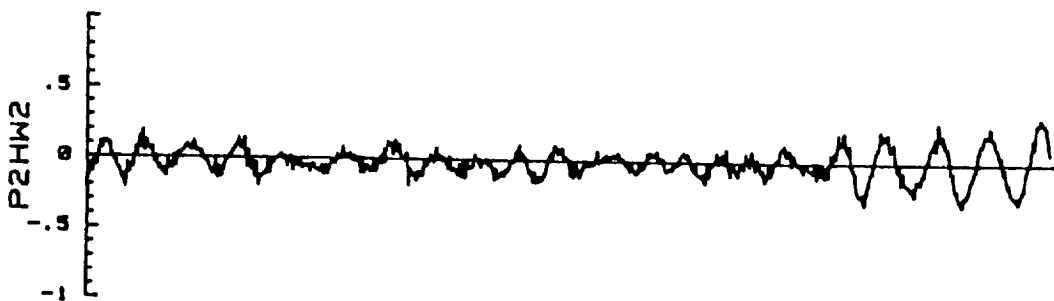
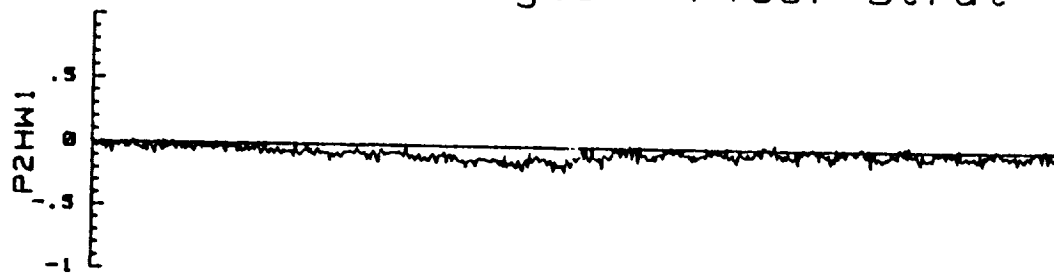
LOG FILE 934REDUCE

	TUNNEL CONDITIONS	LOCAL WALL PROBE CONDITIONS	LOCAL FLOOR PROBE CONDITIONS
Mach	.8001725		
Reynolds No.	9.79489233333		
Pt	709.55	708.603433333	710.075833333
Ps	465.4	477.4245	475.0303
Tt	80.3028133333	540.302813333	540.302813333
Velocity	858.406598442	832.201491334	839.235011328
Density	.018230240804	.0185583708825	.0185028169039
LOG(RhoU)		1.18876833942	1.19112144477
MEAN(HW1)		4.82403933333	5.75614
MEAN(HW2)		4.67863933333	4.93762666667
MEAN(HW3)		5.40758666667	5.3251
S(U) (HW1)		.0801895340403	.0711717036965
S(Rho)(HW1)		.246631035249	.151745706072
S(To) (HW1)		-.371686324654	-.217170651926
S(U) (HW2)		.0815796208448	.0114987894883
S(Rho)(HW2)		.22379374977	.179632925078
S(To) (HW2)		-.798782032582	-.724845767689
S(U) (HW3)		.0757205995021	.0298907458578
S(Rho)(HW3)		.231703194152	.194507190167
S(To) (HW3)		-.523546487225	-.427212086868
u'/U (rms)		.0132928317592	-9.99999999999E+6
p'/P (rms)		.00400499662361	-9.99999999999E+6
to'/To (rms)		.000224062285135	-9.99999999999E+6
R(RhoU)		-.997767275712	-9.99999999999E+6
R(UT0)		.872062983427	-9.99999999999E+6
R(RhoT0)		-.875770964388	-9.99999999999E+6
M'/M		.00930062427607	-9.99999999999E+6
P'/P		.00725524205695	-9.99999999999E+6

Figure 5. Observation Report

8FT Hotwire Voltages - Floor Strut 16 Nov 1987

TEST 934.
RUN 8.
POINT 3.
TIME 11:22:56
DATE 11/16/87
Ps 0.178900E+04
Pt 0.199850E+04
Tt 0.781682E+02
MACH 0.400911E+00
REYNO 0.169156E+02



Scalars have NOT
been applied to this data

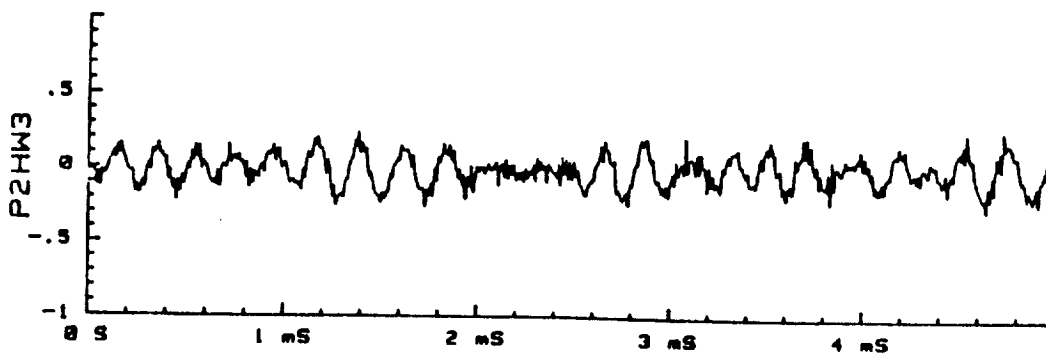


Figure 6. Plot: Waveforms - Typical time trace

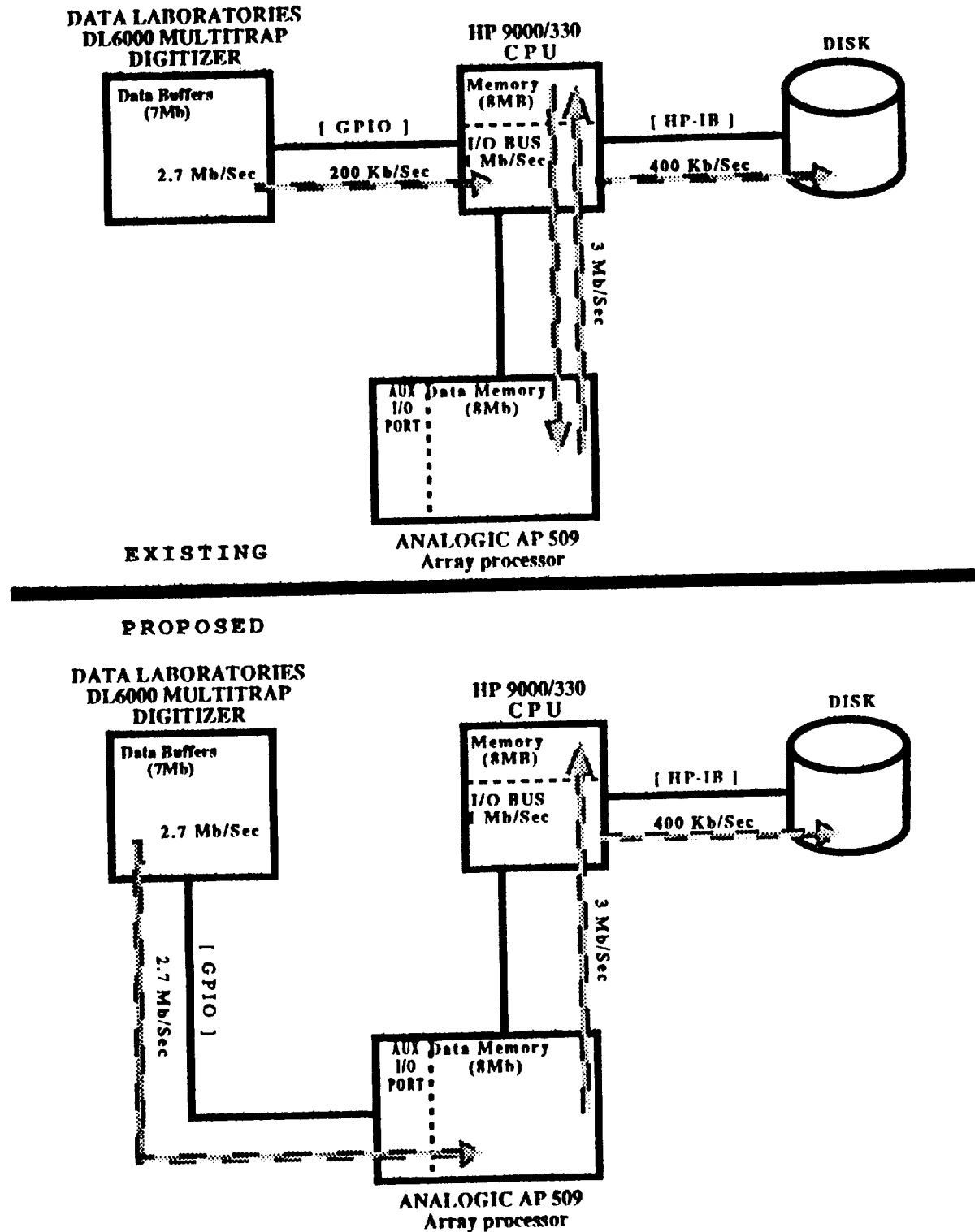


Figure 7. Bottleneck fix - a Proposal

< blank page >

Amplifier/filters Inc.	System 6000	Precision Filters,
High Pass filter (6 pole): 8ch	6622-B-HP1-6ACD	Precision Filters, Inc.
Low Pass filter (6 pole): 8ch	6622-B-LP1-6ACDGO	Precision Filters, Inc.
Pre amplifier: 4ch	6201-2-AMP-CM	Precision Filters, Inc.
High Pass Filter (8 pole): 4ch	6612-C-HP8-6CDM	Precision Filters, Inc.
Low Pass filter (8 pole): 4ch	6612-A-LP8-46CDM	Precision Filters, Inc.
Post amplifier: 4ch	6614-1-AMP-46CM-M101	Precision Filters, Inc.
Calibration module		Precision Filters, Inc.
Array processor	AP509-1024-04-12-01	Analogic, Inc.
Includes: 1 M word data memory 2 auxiliary I/O ports		
Computer	9000/330	Hewlett-Packard
Includes:		
RS232 serial port		
Disk interface		
Thin LAN		
4 Mb memory		
DMA		
Floating point processor		
Audio output		
CPU options:		
monitor	35741A	Hewlett-Packard
video subsystem	98543A	Hewlett-Packard
keyboard	46021A	Hewlett-Packard
memory (4 Mb)	EM300	Infotek
storage devices:		
1 Mb disk (3.5" floppy)	9122C	Hewlett-Packard
20 Mb disk/67Mb tape	7942	Hewlett-Packard
55 Mb disk	7945	Hewlett-Packard
650 Mb optical disk	5300	IEM
340 Mb disk	H5-HP300H	IEM
1600 bpi 9track tape	7475	Hewlett-Packard
I/O interface cards		
HPiB	98624A	Hewlett-Packard
GPIO	98622A	Hewlett-Packard
LAN	98643A	Hewlett-Packard
BCD	98623A	Hewlett-Packard
expansion chassis	98570A	Hewlett-Packard
expansion chassis	98568A	Hewlett-Packard
Data scopes (14ea)		Datacheck
Digitizer	Multitrap d16000	Data Laboratories, Ltd.
Digitizer timebase	d16010	Data Laboratories, Ltd.
Digitizer modules:		
1 M Samples/sec (1ch) 6ea	d16022	Data Laboratories, Ltd.
[256K sample buffer per ch]		
256 K Samp./sec (4ch) 2ea	d16025	Data Laboratories, Ltd.
[256K sample buffer per ch]		
Plotter	7550	Hewlett-Packard
Printer	2934A	Hewlett-Packard
Signal Monitor Selector (2ea)	- -	NASA LaRC
Time Code Generator/Translator	9310	Datum
Voltmeter		Hewlett-Packard
FM tape recorder	96	Honeywell
28 track (Wideband I)		

Table 1. Equipment list

Pkt\$(1) <stx>	(ignored)	
(2)	TEST	
(3)	RUN	
(4)	POINT	
(5)	TIME	
(6)	DATE	
(7)	Ps	tunnel static pressure (psf)
(8)	Pt	tunnel total pressure (psf)
(9)	Tt	Tunnel total temperature (deg F)
(10)	Mach number	
(11)	Reynolds number (per chord foot)	
(12)	PtS1	Strut 1 (Wall) total pressure (psf)
(13)	PsS1	static pressure (psf)
(14)	TtS1	total temperature (deg F)
(15)	PtS2	Strut 2 (Floor) total pressure (psf)
(16)	PsS2	static pressure (psf)
(17)	TtS2	total temperature (deg F)
(18)	PtS3	Strut 3 (Unused)
(19)	PsS3	
(20)	TtS3	
(21)	P1HW1	Strut 1 Hot wire 1 mean voltage
(22)	P1HW2	2 mean voltage
(23)	P1HW3	3 mean voltage
(24)	P1HW4	4 mean voltage
(25)	P2HW1	Strut 2 Hot wire 1 mean voltage
(26)	P2HW2	2 mean voltage
(27)	P2HW3	3 mean voltage
(28)	P3HW1	Strut 3 Hot wire 1 mean voltage
(29)	P4HW1	Strut 4 Hot wire 1 mean voltage
(30)	P5HW1	Strut 5 Hot wire 1 mean voltage
(31)	Kulitel	Microphone RMS voltage
(32)	2	
(33)	3	
(34)	4	
(35)	5	
(36)	6	
(37)	HW1-4GAIN	Gain code representing instrument gain
(38)	HW5GAIN	
(39)	HW6GAIN	Wires 5, 6, and 7 are on probe 2,
(40)	HW7GAIN	wires 1, 2 and 3
(41)	KulitelGAIN	Gain code representing instrument gain
(42)	2	
(43)	3	
(44)	4	
(45)	5	
(46)	6	
(47)	<ETX>	(ignored)

Table 2. Packet Format for 8'TPT test

Pkt\$(1)	<stx>	(ignored)
(2)	TEST	
(3)	RUN	
(4)	POINT	
(5)	Ps	tunnel static pressure (psi)
(6)	Pt	tunnel total pressure (psi)
(7)	Tt	Tunnel total temperature (deg F)
(8)	Mach number	
(9)	Reynolds number	
(10)	Probe Position	(vertical - mm)
(11)	<ETX>	(ignored)

Table 3. Packet Format for 4" pipe flow test

Fluctuating Data File Name Format (voltages)

character	1	
R		Real time digitized data
P		Playback (FM tape) digitized data
character	2	
A		"A" 3-wire probe
B		"B" 3-wire probe
C		"C" 3-wire probe
D		"D" 3-wire probe
character	3,4	
rr		run number (00-99)
character	5,6	
pp		point number(00-99)
character	7,8	
cc		channel number (00-07)
character	9,10	
ss		sequence number - if any assigned
example:	RA171203	real time, probe A, run 17, point 12, no sequence number assigned

The ACQUIRE software module MODUSR2 has been written to incorporate function SET FILE NAMES to utilize the fluctuating data file naming conventions described above.

The ACQUIRE software module MODGEN (provided by Data Laboratories) was modified to not automatically add a sequence number in column 9 if not necessary to differentiate between two files with the same name (in columns one through eight).

Table 4. Fluctuating Data File Name Format (voltages)

<u>Fluctuating Data File Name Format (computed)</u>		
character	1	
V	Velocity	(u'/U) in percent
D	Density	(ρ'/ρ) in percent
T	T51+5emperature	(T_0/T_0) in percent
character	2	
A	"A"	3-wire probe
B	"B"	3-wire probe
C	"C"	3-wire probe
D	"D"	3-wire probe
character	3,4	
rr	run number	(00-99)
character	5,6	
pp	point number	(00-99)
character	78	
—	two underscores	
character	9,10	
ss	sequence number	- if any assigned
example:	VA1712__2	Velocity Probe A, run 17, point 12, second sequence number assigned (third file)

The ACQUIRE software module MODUSR2 has been written to incorporate function SET F FILE NAMES to utilize the fluctuating data file naming conventions described above.

The ACQUIRE software module MODGEN (provided by Data Laboratories) was modified to not automatically add a sequence number in column 9 if not necessary to differentiate between two files with the same name (in columns one through eight).

Table 5. Fluctuating Data File Name Format (computed)

< blank page >

APPENDICES

PRECEDING PAGE BLANK NOT FILMED

~~78~~ 78 INTENTIONALLY BLANK

< blank page >

APPENDIX A. Function definitions

This appendix contains the DDAPS functions added to ACQUIRE by the user to provide the necessary functionality to acquire, process, display, store and transmit the hot wire data.

The function definition sheets provide essential definition data for each function. Full documentation for each function is contained within the program source code listings.

These routines were written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC, FldMD/EMB under contract NAS1 - 18585, Task 42. This work began in October, 1986 and continues thru October, 1990.

< blank page >

AP CONST ARITH

Function Name: AP CONST ARITH
Function Number: 1507
Module: MOD7

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[AP CONST ARITH]-----|

AP CONST K

Function Name: AP CONST K
Function Number: 1501
Module: MOD7

This function specifies the value of the constant that will be used with the array processor constant arithmetic functions. The constant is saved in an internal buffer, and is also sent to the Analogic AP509 Array Processor.

```
[ AP CONST K ]-----|-----|
                    |
                    |--[ = ]--[ value ]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
value +1E100	numeric real	-1.E100 to

EXAMPLE

AP CONST K = .5

AP EVALUATE

Function Name: AP EVALUATE
Function Number: 1504
Module: MOD7

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[AP EVALUATE]-----|

AP EVALUATE MORE

Function Name: AP EVALUATE MORE
Function Number: 1550
Module: MOD7

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[AP EVALUATE MORE]-----|

AP INIT

Function Name: AP INIT
Function Number: 1570
Module: MOD7

This function INITIALIZES the Analogic AP509 Array Processor. The AP is first reset, and then the array processor program is transferred from disk to the array processor.

This function is automatically invoked when the program is first executed (Power On), but needs to be manually invoked after the keyboard RESET is pressed, since the AP is also reset via the I/O system of the computer. Unless the AP is re-initialized after a system reset, AP functions will hang the system - the AP program is lost when reset is detected.

[AP INIT]-----|

AP K+MEM

Function Name: AP K+MEM
Function Number: 1526
Module: MOD7

This function will add the constant K to every data value between the two cursors on the selected trace. The new calculated data overwrites the old data. The value of constant K is set by function AP CONST K.

ALGORITHM

$$X_i = X_i + K$$

for i=C1 to C2

Where:

X_i is the data value at address i, in Y calibrated units

C1, C2 are the beginning and ending cursor position addresses

[AP K+MEM]-----|
 | |
 |--(trace)-----|

<u>Item</u>	<u>Description</u>	<u>Range</u>
trace	Integer	1, 2, 3

AP K-MEM

Function Name: AP K-MEM
 Function Number: 1527
 Module: MOD7

This function will subtract every data value between the two cursors on the selected trace from the constant K. The new calculated data overwrites the old data. The value of constant K is set by function AP CONST K.

ALGORITHM

$$X_i = K - X_i$$

for i=C1 to C2

Where:

X_i is the data value at address i, in Y calibrated units
 C1, C2 are the beginning and ending cursor position addresses

[AP K-MEM] -----
 |
 | --(trace)-----
 |

Item	Description	Range
trace	Integer	1, 2, 3

NOTE

For MEM-K, set constant K as -K, and use function K+MEM.

AP K*MEM

Function Name: AP K*MEM
Function Number: 1528
Module: MOD7

This function will multiply every data value between the two cursors on the selected trace by the constant K. The new calculated data overwrites the old data. The value of constant K is set by function AP CONST K.

ALGORITHM

$$X_i = K * X_i$$

for i=C1 to C2

Where:

X_i is the data value at address i, in Y calibrated units

C1, C2 are the beginning and ending cursor position addresses

[AP K*MEM]-----|
 | |
 |--(trace)-----|

Item	Description	Range
trace	Integer	1, 2, 3

AP K/MEM

Function Name: AP K/MEM
Function Number: 1529
Module: MOD7

This function will divide every data value between the two cursors on the selected trace into the constant K. The new calculated data overwrites the old data. The value of constant K is set by function AP CONST K.

ALGORITHM

$$X_i = \frac{K}{X_i}$$

for i=C1 to C2

Where:

X_i is the data value at address i, in Y calibrated units
C1, C2 are the beginning and ending cursor position addresses

[AP K/MEM]-----|
 |
 |---(trace)-----|

Item	Description	Range
trace	Integer	1, 2, 3

NOTE:

For MEM/K, set constant K as $\frac{1}{K}$, and use function K*MEM.

AP K:MEM

Function Name: AP K:MEM
Function Number: 1530
Module: MOD7

This function will make every data value between the two cursors on the selected trace equal to the constant K. The new calculated data overwrites the old data. The value of constant K is set by function AP CONST K.

ALGORITHM

$X_i = K$
for i=C1 to C2

Where:

X_i is the data value at address i, in Y calibrated units

C1, C2 are the beginning and ending cursor position addresses

[AP K:MEM]-----|
 | |
 |---(trace)-----|

Item	Description	Range
trace	Integer	1, 2, 3

AP MEM A+MEM B

Function Name: AP MEM A+MEM B
Function Number: 1531
Module: MOD7

This function will add every data value between the two cursors on trace 1 to every data value between the cursors on trace 2. The new calculated data overwrites the old data on trace 1.

ALGORITHM

$$X_i = X_i + Y_j$$

for i=C1 to C2, and j=C3 to C4

Where: X_i is the data value at address i, in Y calibrated units
 Y_j is the data value at address j, in Y calibrated units

[AP MEM A+MEM B]-----|

AP MEM A-MEM B

Function Name: AP MEM A-MEM B
Function Number: 1532
Module: MOD7

This function will subtract every data value between the two cursors on trace 2 from every data value between the cursors on trace 1. The new calculated data overwrites the old data on trace 1.

ALGORITHM

$$X_i = X_i - Y_j$$

for i=C1 to C2, and j=C3 to C4

Where: X_i is the data value at address i, in Y calibrated units
 Y_j is the data value at address j, in Y calibrated units

[AP MEM A-MEM B]-----|

AP MEM A*MEM B

Function Name: AP MEM A*MEM B
Function Number: 1533
Module: MOD7

This function will multiply every data value between the two cursors on trace 2 with every data value between the cursors on trace 1. The new calculated data overwrites the old data on trace 1.

ALGORITHM

$$X_i = X_i * Y_j$$

for i=C1 to C2, and j=C3 to C4

Where: X_i is the data value at address i, in Y calibrated units
 Y_j is the data value at address j, in Y calibrated units

[AP MEM A*MEM B]-----|

AP MEM A/MEM B

Function Name: AP MEM A/MEM B
Function Number: 1534
Module: MOD7

This function will divide every data value between the two cursors on trace 1 by every data value between the cursors on trace 2. The new calculated data overwrites the old data on trace 1.

ALGORITHM

$$X_i = X_i / Y_j$$

for i=C1 to C2, and j=C3 to C4

Where: X_i is the data value at address i, in Y calibrated units
 Y_j is the data value at address j, in Y calibrated units

[AP MEM A/MEM B]-----|

AP MEM A:MEM B

Function Name: AP MEM A:MEM B
Function Number: 1535
Module: MOD7

This function copys every data value between the two cursors on trace 2 to every data value between the cursors on trace 1.

ALGORITHM

$X_i = Y_j$
for $i=C1$ to $C2$, and $j=C3$ to $C4$

Where: X_i is the data value at address i , in Y calibrated units
 Y_j is the data value at address j , in Y calibrated units

[AP MEM A:MEM B]-----|

AP MANIPULATE

Function Name: AP MANIPULATE
Function Number: 1505
Module: MOD7

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[AP MANIPULATE]-----|

AP MATRIX

Function Name: AP MATRIX
Function Number: 1590
Module: MOD7

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[AP MATRIX]-----|

AP MAX

Function Name: AP MAX
Function Number: 1516
Module: MOD7

This function will calculate the maximum data value between the two cursors on the selected trace.

[AP MAX]-----|
 |
 |---(trace)-----|

<u>Item</u>	<u>Description</u>	<u>Range</u>
trace	Integer	1, 2, 3

AP MIN

Function Name: AP MIN
Function Number: 1517
Module: MOD7

This function will calculate the minimum data value between the two cursors on the selected trace.

[AP MIN]-----|
 | |
 |--(trace)-----|

<u>Item</u>	<u>Description</u>	<u>Range</u>
trace	Integer	1, 2, 3

AP MEAN

Function Name: AP MEAN
Function Number: 1511
Module: MOD7

This function will calculate the mean data value between the two cursors on the selected trace.

ALGORITHM

$$\frac{\sum_{i=C1}^{i=C2} X_i}{n}$$

Where: X_i is the data value at address i , in Y calibrated units
 n is the number of points ($C2-C1+1$)
 $C1, C2$ are the beginning and ending cursor position addresses

[AP MEAN] -----
 | |
 |---(trace)-----|

<u>Item</u>	<u>Description</u>	<u>Range</u>
trace	Integer	1, 2, 3

AP MEM ARITH

Function Name: AP MEM ARITH
Function Number: 1506
Module: MOD7

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[AP MEM ARITH]-----|

AP MENU

Function Name: AP MENU
Function Number: 1500
Module: MOD7

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[AP MENU]-----|

AP REMOVE MEAN

Function Name: AP REMOVE MEAN
 Function Number: 1524
 Module: MOD7

This function will first calculate the mean data value between the two cursors on the selected trace, and then subtract that value from every data value between the two cursors on the selected trace. The new calculated data overwrites the old data.

ALGORITHM

$$\text{MEAN} = \frac{\sum_{i=C1}^{i=C2} X_i}{n}$$

$$X_i = X_i - \text{MEAN}$$

Where: X_i is the data value at address i , in Y calibrated units
 n is the number of points ($C2-C1+1$)
 $C1, C2$ are the beginning and ending cursor position addresses

[AP REMOVE MEAN] -----
 |
 | --(trace)----- |

Item	Description	Range
trace	Integer	1, 2, 3

AP RMS

Function Name: AP RMS
 Function Number: 1513
 Module: MOD7

This function will calculate the root mean square of the data values between the two cursors on the selected trace.

ALGORITHM

$$\sqrt{\frac{\sum_{i=C1}^{i=C2} X_i^2}{n}}$$

Where: X_i is the data value at address i , in Y calibrated units
 n is the number of points ($C2-C1+1$)
 $C1, C2$ are the beginning and ending cursor position addresses

[AP RMS]-----|
 |
 |---(trace)-----|

Item	Description	Range
trace	Integer	1, 2, 3

AP SDEV

Function Name: AP SDEV
 Function Number: 1512
 Module: MOD7

This function will calculate the standard deviation of the data values between the two cursors on the selected trace.

ALGORITHM

$$\sqrt{\left(\frac{\sum_{i=C1}^{i=C2} X_i^2}{n} \right) - \left(\frac{\sum_{i=C1}^{i=C2} X_i}{n} \right)^2}$$

Where: X_i is the data value at address i , in Y calibrated units
 n is the number of points ($C2-C1+1$)
 $C1, C2$ are the beginning and ending cursor position addresses

[AP SDEV]-----|
 |
 |--(trace)-----|

Item	Description	Range
trace	Integer	1, 2, 3

AP SMOOTH

Function Name: AP SMOOTH
Function Number: 1519
Module: MOD7

This function will perform a 4 point smooth on the data between the two cursors on the selected trace, and overwrite the old data with the new calculated data.

[AP SMOOTH]-----|
 | |
 |--(trace)-----|

<u>Item</u>	<u>Description</u>	<u>Range</u>
trace	Integer	1, 2, 3

AP SUM OF SQ

Function Name: AP SUM OF SQ
Function Number: 1514
Module: MOD7

This function will calculate the sum of the square of the data values between the two cursors on the selected trace.

ALGORITHM

$$\sum_{i=C1}^{i=C2} X_i^2$$

Where: X_i is the data value at address i , in Y calibrated units
 n is the number of points ($C2-C1+1$)
 $C1, C2$ are the beginning and ending cursor position addresses

[AP SUM OF SQ]-----|
 | |
 |---(trace)-----|

<u>Item</u>	<u>Description</u>	<u>Range</u>
trace	Integer	1, 2, 3

AP TRANSFER

Function Name: AP TRANSFER
Function Number: 1580
Module: MOD7

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[AP TRANSFER]-----|

This function transfers data from the cpu to the array processor. When invoked by the operator, the data between the two cursors in the selected trace is transferred to the respective buffer in the array processor - trace 1 to buffer 1, etc. This function is not normally invoked by the operator, since its use is implicit in other array processor functions (ie: AP RMS, AP K*MEM, etc.). Programmatically, an internal variable, Auto_xfer_on, when set equal to 0, inhibits the automatic invocation of the transfer functions. This function could then be programmatically invoked once at the beginning of a sequence of array processor functions, saving the time to transfer intermediate computational results to (and from) the array processor. It is important, of course, to invoke AP TO CPU to retrieve the results from the array processor, and to reset Auto_xfer_on to 1.

```
[ AP TF FROM CPU ] -----
```

```
|                                |
```

```
| --(trace)--                  |
```

```
|                                |
```

Item	Description	Range
trace	Integer	1, 2, 3

AP TF MAP CHAN

Function Name: AP TF MAP CHAN
Function Number: 1582
Module: MOD7

This function was intended to map a trace to an internal buffer of the array processor during data transfers. This function does nothing. Since the array processor memory is limited to 1 million samples, the structure can hold only 3 traces (each 256K samples), plus some scratch data areas. Normally, array processor functions operate either on trace 1, 2, or 3, or on traces 1 and 2. The matrix function SIMULT EQ operates with all three traces (1, 2, and 3). In each case, mapping is 1-to-1.

[AP TF MAP CHAN]-----|

This function transfers data to the cpu from the array processor. When invoked by the operator, the data buffer in the array processor is transferred to the selected trace between the two cursors - buffer 1 to trace 1, etc. This function is not normally invoked by the operator, since its use is implicit in other array processor functions (ie: AP MEM A+MEM B, AP K*MEM, etc.). Programmatically, an internal variable, *Auto xfer on*, when set equal to 0, inhibits the automatic invocation of the transfer functions. AP TF FROM CPU could then be programmatically invoked once at the beginning of a sequence of array processor functions, saving the time to transfer intermediate computational results to (and from) the array processor. It is important, of course, to invoke this function to retrieve the results from the array processor, and to reset *Auto xfer on* to 1.

[illegible]

Item	Description	Range
trace	Integer	1, 2, 3

BEEP

Function Name: BEEP
Function Number: 4060
Module: MODUSR1

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[BEEP]-----|

CALC NUSSELT

Function Name: CALC NUSSELT
 Function Number: 4121
 Module: MODUSR2

This function computes and stores non-dimensional numbers such as Nusselt number (Nu), Reynolds number (Re), Knudsen number (Kn), and temperature parameters (θ and τ). These values are necessary in sensitivity algorithms involving Baldwin and/or Morkovin methodology which are implemented in CALC SENS. Observations in the range of INITIAL OBS to ENDING OBS are processed. Computations are repeated for all three wires of the CURRENT PROBE.

Initially, recovery factor (η) is computed from the raw data. A least-square technique is used to fit a curve through computed recovery factors. Function LEAST SQUARE implements this algorithm. Reference hotwire resistances R_{ref} and R_w are retrieved from the logfile for each pressure. For R_{ref} , the logfile is searched for the observation where $M_\infty = .05$ and $T_0 = 80^\circ\text{F}$. Similarly, for R_w , at $M_\infty = .05$ and $T_0 = 120^\circ\text{F}$ at each pressure. Using these values, the thermal coefficient of expansion (α) is computed.

Nusselt number is computed.

$$Nu = \frac{E I}{\eta L k_t (T_w - \eta T_o)}$$

Knudsen number is computed.

$$Kn = \frac{\lambda}{d_w}$$

Reynolds number based on wire diameter is computed:

$$Re = \frac{\rho_\infty U_\infty d}{\mu}$$

The computed numbers are stored in the appropriate probe file for each appropriate observation.

NOTE:
 This function is currently under development.

[CALC NUSSELT]-----|

CALC VEL etc

Function Name: CALC VEL etc
Function Number: 4104
Module: MODUSR2

This function computes velocity, density and temperature fluctuations as ratios of the fluctuating quantities to the mean quantities represented as a percentage: u'/U ρ'/ρ t'/T

The log file for each observation in the range of INITIAL OBS to ENDING OBS is processed for each probe in the range of INITIAL PROBE to ENDING PROBE. For each observation, and each probe, the three fluctuating data files related to the Run and Point (variables 2 and 3 in the observation record) are loaded into traces (and memories) 1, 2, and 3. See function SET FILE NAMES.

The computations retrieve mean values, sensitivities, and gains related to the data in channels 1, 2 and 3. For each simultaneous sample in each of the three traces (the beginning and ending samples are defined by the cursors on trace 1) the instantaneous value is divided by the equivalent mean value and the gain. The three ratios are then matrix multiplied by the matrix inversion of the sensitivities. This process essentially solves a set of simultaneous equations for three unknowns: u'/U , ρ'/ρ , and t'/T_0 .

The solutions replace the traces (memories) 1, 2 and 3. The mean value is removed, and the RMS (root mean square) value of each of the answers is stored in the logfile for the appropriate observation and probe. Traces 1, 2, and 3 are then stored in separate disk files in the disk defined by function DATA DISC DEV. The disk file names are generated by function SET F FILENAMES.

NOTE:

This function can take as long as 80 minutes to complete (without the Array Processor) for 256K samples per wire. However, the array processor reduces the time to less than 2 minutes of computing time.

[CALC VEL etc]-----|

CATALOG

Function Name: CATALOG
Function Number: 4033
Module: MODUSR1

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[CATALOG]-----|

CAT GROUP

Function Name: CAT GROUP

Function Number: 4021

Module: MODUSR1

This function catalogs all selected files. The selection criteria is defined in detail in the HP BASIC language reference manual for HP function CAT (SELECT).

```
[ CAT GROUP ]-----|
                    |
                    |--[ - ]--[ name ]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
name	string expression	any valid characters that are allowed in a file name

CODE TO GAINS

Function Name: CODE TO GAINS
Function Number: 4125
Module: MODUSR2

This function computes actual voltage gains from gain codes. The fluctuating data is gained to provide adequate voltage for filtering and digitization, and the gain codes are sent in a data packet from the static data computer during the logging of a data point. For the probes in the range INITIAL PROBE to ENDING PROBE, and for observations in the range INITIAL OBS to ENDING OBS, the gain codes are retrieved from the observation file, and thru a table lookup algorithm, the actual gains are retrieved, and stored back into the appropriate place in the logfile for the probe and observation being processed.

NOTE: This functions use is restricted to early logfiles, where gain codes, rather than actual gains were initially logged. Once the codes are converted and gains written over the codes, this function becomes obsolete.

[CODE TO GAINS]-----|

COEF FILENAME

Function Name: COEF FILENAME
Function Number: 4101
Module: MODUSR2

This function selects the file name relaven to the hotwire coefficient file being processed. See functions LOAD COEFS and STORE COEFS.

[COEF FILENAME]-----|

COMPUTE COEFS

Function Name: COMPUTE COEFS
Function Number: 4123
Module: MODUSR2

This function computes hotwire coefficients for the probe previously selected - by function CURRENT PROBE - utilizing a custom multiple linear regression routine that generates up to 10 coefficients based on calibration data already in the logfile. These coefficients are then stored - by function STORE COEFS - in a coefficient file whose name has been previously defined - by function COEF FILENAME. Function COMPUTE SENS utilizes these coefficients to generate sensitivities necessary to compute velocity, density and temperature turbulence ratios - by function CALC VEL etc.

NOTE:

This function has not been implemented. SUB Mlr in the software has been created, but the implementation is not complete. Mlr is an adaptation of the Mlr routine in the HP 98820A package described below.

The logfiles can be processed by a separate program (as though they were created by the statistical program). Use of the HP Statistical Library - 98820A - is recommended. It resides on disk as file STATS. The statistics package is complimentary to ACQUIRE, but is not part of it.

[COMPUTE COEFS]-----|

COMPUTE R etc

Function Name: COMPUTE R etc
Function Number: 4132
Module: MODUSR2

This function computes the correlations between velocity, density and temperature fluctuations and then computes mass flow fluctuation $[m'/m(rms)]$ and pressure fluctuation $[p'/P(rms)]$ using the correlation between velocity, density, and temperature fluctuations.

The log file for each observation in the range of INITIAL OBS to ENDING OBS is processed for each probe in the range of INITIAL PROBE to ENDING PROBE.

For each observation, and each probe, the three fluctuating data files related to the Run and Point (variables 2 and 3 in the observation record) are loaded into traces (and memories) 2, 3, and 4.

As each pair of traces is multiplied together, the resulting trace ends up in trace 1. The correlation of the two traces multiplied is the ratio of the rms value of the first trace multiplied by the rms value of the second trace to the mean value of trace 1. The rms values of the first and second traces have been previously calculated by function CALC VEL etc, and were called u'/U , p'/P , and t'/T_0 .

The correlation between these three components of turbulence are stored in the appropriate observation record for the observation and probe being processed as $R(RhoU)$, $R(UT_0)$, and $R(RhoT_0)$.

Massflow and pressure fluctuations are then computed from the correlations just computed, and these are also stored in the appropriate logfile as m'/m and p'/P .

[COMPUTE R etc]-----|

COMPUTE SENS

Function Name: COMPUTE SENS

Function Number: 4111

Module: MODUSR2

This function computes hotwire sensitivities for the current probe as defined by CURRENT PROBE for all observations in the range INITIAL OBS to ENDING OBS. The hotwire coefficients previously defined - see functions GET COEFS, COEF FILENAME, LOAD COEFS, ENTER COEFS, EDIT COEFS, STORE COEFS, AND COMPUTE COEFS - are utilized to compute sensitivities utilizing a variety of algorithms. The algorithm executed by this function must have been previously selected by function SENS ALG. These algorithms are:

- 1 - based on $E = f(U, \rho, T_0)$ in LOG space
- 2 - based on $Nu = f(M, Kn, \tau)$ in LOG space (Baldwin)
- 3 - based on $Nu = f(M, Re, \theta)$ in LOG space (Morkovin)
- 4 - based on $E = f(U, \rho, T_0)$ POLY
- 5 - based on $Nu = f(M, Kn, \tau)$ POLY (Baldwin)
- 6 - based on $Nu = f(M, Re, \theta)$ POLY (Morkovin)

The computed sensitivities are stored in the appropriate probe file for each appropriate observation.

These sensitivities are read from the logfile - by function CALC VAL etc - to compute velocity, density and temperature turbulence ratios.

NOTE:

The computed sensitivities overwrite any previously computed sensitivities.

[COMPUTE SENS]-----|

CURRENT OBS

Function Name: CURRENT OBS
Function Number: 4114
Module: MODUSR2

This function declares the current observation to be processed by other MODUSR2 functions. Some functions utilizing this feature to define a specific observation to be processed are START TIME, END TIME, and SET FILE NAMES.

```
[ CURRENT OBS ]-----|  
                    |  
                    |--[ - ]--[value]--|
```

Item	Description	Range
value	numeric integer	INITIAL OBS to ENDING OBS

CURRENT PROBE

Function Name: CURRENT PROBE
Function Number: 4132
Module: MODUSR2

This function declares the current probe to be processed by other MODUSR2 functions. Some functions utilizing this feature to define a specific probe to be processed are START TIME, END TIME, and SET FILE NAMES.

```
[ CURRENT PROBE ]-----|  
|                         |  
|--[ - ]--[value]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
value	numeric integer	INITIAL PROBE to ENDING PROBE

DIGITIZE ENABLE

Function Name: DIGITIZE ENABLE
Function Number: 4052
Module: MODUSR1

This function sets an internal flag (in MODUSR2). This flag also returns the value of the flag, if no parameter is specified when invoking the function. Use of this function in a sequence program can alter the sequence program process, depending on the value of the flag.

Internally, function LOG DATA POINT test the sense of the flag, and if true (1), assumes that fluctuating data is available in mems 1, 2, and 3. It then computes the RMS values of the three hotwire fluctuating voltages.

In a sequence program, the sense of the flag can be put into a sequence variable, and tested. This technique would then selectively enable a part of the sequence program involved in dynamic data digitization - including the storage of the dynamic data files on disk.

```
[ DIGITIZE ENABLE ]-----|
                        |
                        |--[- 0]-----|
                        |--[- 1]-----|
```

where 0 - Disable
1 - Enable

EDIT COEFS

Function Name: EDIT COEFS
Function Number: 4133
Module: MODUSR2

This function allows the operator to manually edit hot wire calibration coefficients (up to 10) for three wires. These coefficients have been previously generated. The operator views a list of the entered coefficients, and, by responding to prompts, select the coefficient to be edited. After each coefficient is edited, the operator may accept the coefficients, or be prompted to select another coefficient to be edited. The operator should then invoke function STORE COEFS.

[EDIT COEFS]-----|

NOTE: This function is highly interactive, and is not recommended for inclusion in a SEQUENCE PROGRAM.

END TIME

Function Name: END TIME
Function Number: 4137
Module: MODUSR2

This function returns the FM tape stop time for the current observation - according to the log file - and also retrieves search limit times based on the initial observation start time and the ending observation stop time. This function is used to determine the stop times during playback functions. The time returned is a real number representing the number of seconds since the beginning of the year. Day of year information is not encoded in this time.

This function is used internally by various functions, and is rarely invoked by the operator.

[END TIME]-----|

ENDING OBS

Function Name: ENDING OBS
Function Number: 4113
Module: MODUSR2

This function declares the ending observation to be processed by other MODUSR2 functions - see function INITIAL OBS. Functions utilizing this feature to define the range of observations to be processed include: COMPUTE SENS, LOG DATA POINT, CODE TO GAINS, CALC VEL etc, COMPUTE R etc, Remake Probe, and LOGFILE TO PC.

```
[ ENDING OBS ]-----|
                  |
                  |--[ - ]--[value]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
value	numeric integer	1 to 300, the max number of observations

ENDING PROBE

Function Name: ENDING PROBE
Function Number: 4131
Module: MODUSR2

This function declares the ending probe to be processed by other MODUSR2 functions - see function INITIAL PROBE. Functions utilizing this feature to define the range of probes to be processed are: COMPUTE SENS, LOG DATA POINT, CODE TO GAINS, CALC VEL etc, COMPUTE R etc, Remake probe and LOGFILE TO PC.

```
[ ENDING PROBE ]-----|
                    |
                    |--[ - ]--[value]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
value	numeric integer	1 to 4, the max number of probes

ENTER COEFS

Function Name: ENTER COEFS
Function Number: 4103
Module: MODUSR2

This function allows the operator to manually enter hot wire calibration coefficients (up to 10) for three wires. These coefficients have been previously generated elsewhere, and are not available for entry via disc (see function LOAD COEFS). The operator is prompted for each coefficient by wire, number, and name. Once all 30 coefficients are entered (unused coefficients should be set to 0.0), the operator views a list of the entered coefficients, and chooses to accept or reject the coefficients. If they are accepted, the function is complete. The operator should then invoke function STORE COEFS. If the coefficients are not accepted - because they are not correct, this function automatically enters the EDIT COEFS function.

[ENTER COEFS]-----|

NOTE: This function is highly interactive, and is not recommended for inclusion in a SEQUENCE PROGRAM.

FILE COPY

Function Name: FILE COPY
Function Number: 4025
Module: MODUSR1

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[FILE COPY]-----|

FILE GROUP

Function Name: FILE GROUP
Function Number: 4028
Module: MODUSR1

This function defines the files to be selected for copying - function COPY FILES - or moving - function MOVE FILES. the files selected will begin with, or be equal to the character(s) defined by this function. For example, if this function defines "ABC", then all of the files in the TO DISK device that begin with "ABC" would be selected for copying or moving. Note that this function only defines the character(s): no selection is done until COPY FILES or MOVE FILES is invoked.

```
[ FILE GROUP ]-----|
|                      |
|  --[ - ]--[ name ]--|
```

Item	Description	Range
name	string expression	any valid characters that are
allowed		in a file name

FILE TRANSFERS

Function Name: FILE TRANSFERS
Function Number: 4126
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[FILE TRANSFERS]-----|

FM AUTOPLAYBACK

Function Name: FM AUTOPLAYBACK
Function Number: 4312
Module: MODUSR4

One of the MOTION category of FM tape functions, invoking this function places the tape in motion in a forward direction. Tape speed will be that selected by function FM TAPE SPEED. At the completion of the playback period, the tape will stop. Both the start time and the stop time will be retrieved from the logfile, and the tape will be positioned by reading the time - as read by the time code generator (must be in the TRANSLATE mode).

The automatic feature of this function is that the tape first locates and then plays the specified data automatically, then stops playing back automatically. The start and stop time of the observation previously selected by function CURRENT OBS defines the desired tape segment for playback.

When placed in a sequence program, this function will hold off further execution of other sequence lines until the start time has been found. Function START TIME FOUND can be used to test if the start time was found (a 1 is returned), or if the tape (and the function) terminated by finding only the stop time. Invoking the FM STOP function will terminate execution of this function.

[FM AUTOPLAYBACK]-----|

WARNING

WARNING

Any tape motion constitutes a hazard to the operator. This motion can be more of a surprise if the commands are remote from the transport.

Assure that the transport door is closed before any tape motion commands are invoked.

Assure that personnel contact with tape reels cannot be accomplished when tape motion can occur.

Take the transport out of remote when loading or unloading tape by pressing any tape speed button on the transport control panel, releasing the remote button. Note that the remote light will remain on to indicate that a remote cable is connected to the transport.

Never engage remote control (never press the remote button) unless the transport door is closed, separating the operator from the tape, and tape reels.

FM AUTORECORD

Function Name: FM AUTORECORD
Function Number: 4311
Module: MODUSR4

One of the MOTION category of FM tape functions, invoking this function places the tape in motion in a forward direction, and enables the record logic in the tape transport. (Note: switches internal to the transport can inhibit recording, overriding the record capability of this function.) Tape speed will be that selected by function FM TAPE SPEED. At the completion of the recording period, the tape will stop. Both the start time and the stop time will be retrieved from the time code generator (must be in the GENERATE mode), and saved in an internal buffer for later logging by function LOG DATA POINT.

The automatic feature of this function is that the tape records for a specified number of seconds as defined by function FM RECORD TIME, then stops recording automatically. When placed in a sequence program, this is a no-wait function. Use of the WAIT ! sequence program command language enhancement will wait for this function to complete.

[FM AUTORECORD]-----|

WARNING

WARNING

Any tape motion constitutes a hazard to the operator. This motion can be more of a surprise if the commands are remote from the transport.

Assure that the transport door is closed before any tape motion commands are invoked.

Assure that personnel contact with tape reels cannot be accomplished when tape motion can occur.

Take the transport out of remote when loading or unloading tape by pressing any tape speed button on the transport control panel, releasing the remote button. Note that the remote light will remain on to indicate that a remote cable is connected to the transport.

Never engage remote control (never press the remote button) unless the transport door is closed, separating the operator from the tape, and tape reels.

FM FORWARD

Function Name: FM FORWARD
Function Number: 4304
Module: MODUSR4

One of the MOTION category of FM tape functions, invoking this function places the tape in motion in a forward direction. The FM tape transport must be in remote, and the tape must be loaded (see function FM STOP for more detail). Tape speed will be that selected by function FM TAPE SPEED.

[FM FORWARD]-----|

WARNING

WARNING

Any tape motion constitutes a hazard to the operator. This motion can be more of a surprise if the commands are remote from the transport.

Assure that the transport door is closed before any tape motion commands are invoked.

Assure that personnel contact with tape reels cannot be accomplished when tape motion can occur.

Take the transport out of remote when loading or unloading tape by pressing any tape speed button on the transport control panel, releasing the remote button. Note that the remote light will remain on to indicate that a remote cable is connected to the transport.

Never engage remote control (never press the remote button) unless the transport door is closed, separating the operator from the tape, and tape reels.

FM MOTION

Function Name: FM MOTION
Function Number: 4320
Module: MODUSR4

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[FM MOTION]-----|

WARNING

WARNING

Any tape motion constitutes a hazard to the operator. This motion can be more of a surprise if the commands are remote from the transport.

Assure that the transport door is closed before any tape motion commands are invoked.

Assure that personnel contact with tape reels cannot be accomplished when tape motion can occur.

Take the transport out of remote when loading or unloading tape by pressing any tape speed button on the transport control panel, releasing the remote button. Note that the remote light will remain on to indicate that a remote cable is connected to the transport.

Never engage remote control (never press the remote button) unless the transport door is closed, separating the operator from the tape and tape reels.

FM POSITION

Function Name: FM POSITION
Function Number: 4315
Module: MODUSR4

One of the MOTION category of FM tape functions, invoking this function positions the tape at the beginning, or first start time of interest - the start time logged for the the initial observation as defined by function INITIAL OBS. The tape will shuttle forwards and backwards, searching for the start time, and will then stop the tape motion, leaving the tape positioned just prior to the initial observation. The tape will be positioned by reading the time - as read by the time code generator (must be in the TRANSLATE mode).

Both the start time and the stop time will be retrieved from the logfile, and the stop time is internally buffered, and is then used by function FM AUTOPLAYBACK to determine absolute search limits when processing start and stop times from observations defined by function CURRENT OBS.

Function START TIME FOUND can be used to test if the start time was found (a 1 is returned), or if the tape (and the function) terminated prior to finding the start time. Invoking the FM STOP function will terminate execution of this function.

[FM POSITION]-----|

WARNING

WARNING

Any tape motion constitutes a hazard to the operator. This motion can be more of a surprise if the commands are remote from the transport.

Assure that the transport door is closed before any tape motion commands are invoked.

Assure that personnel contact with tape reels cannot be accomplished when tape motion can occur.

Take the transport out of remote when loading or unloading tape by pressing any tape speed button on the transport control panel, releasing the remote button. Note that the remote light will remain on to indicate that a remote cable is connected to the transport.

Never engage remote control (never press the remote button) unless the transport door is closed, separating the operator from the tape, and tape reels.

FM RECORD

Function Name: FM RECORD
Function Number: 4302
Module: MODUSR4

One of the MOTION category of FM tape functions, invoking this function places the tape in motion in a forward direction, and enables the record logic in the tape transport. (Note: switches internal to the transport can inhibit recording, overriding the record capability of this function.) Tape speed will be that selected by function FM TAPE SPEED. The start time will be retrieved from the time code generator (must be in the GENERATE mode), and saved in an internal buffer for later logging by function LOG DATA POINT.

[FM RECORD]-----|

WARNING

WARNING

Any tape motion constitutes a hazard to the operator. This motion can be more of a surprise if the commands are remote from the transport.

Assure that the transport door is closed before any tape motion commands are invoked.

Assure that personnel contact with tape reels cannot be accomplished when tape motion can occur.

Take the transport out of remote when loading or unloading tape by pressing any tape speed button on the transport control panel, releasing the remote button. Note that the remote light will remain on to indicate that a remote cable is connected to the transport.

Never engage remote control (never press the remote button) unless the transport door is closed, separating the operator from the tape, and tape reels.

FM RECORD TIME

Function Name: FM RECORD TIME
Function Number: 4301
Module: MODUSR4

When tape motion function FM AUTORECORD is invoked, the recording period is set by this function.

```
[ FM RECORD TIME ]-----|
                    |
                    |--[ - seconds ]-----|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
seconds	numeric integer	
positive integer		any valid

FM REVERSE

Function Name: FM REVERSE
Function Number: 4305
Module: MODUSR4

One of the MOTION category of FM tape functions, invoking this function places the tape in motion in a reverse direction. The FM tape transport must be in remote, and the tape must be loaded (see function FM STOP for more detail). Tape speed will be that selected by function FM TAPE SPEED.

[FM REVERSE]-----|

WARNING

WARNING

Any tape motion constitutes a hazard to the operator. This motion can be more of a surprise if the commands are remote from the transport.

Assure that the transport door is closed before any tape motion commands are invoked.

Assure that personnel contact with tape reels cannot be accomplished when tape motion can occur.

Take the transport out of remote when loading or unloading tape by pressing any tape speed button on the transport control panel, releasing the remote button. Note that the remote light will remain on to indicate that a remote cable is connected to the transport.

Never engage remote control (never press the remote button) unless the transport door is closed, separating the operator from the tape, and tape reels.

FM REWIND

Function Name: FM REWIND
Function Number: 4321
Module: MODUSR4

One of the MOTION category of FM tape functions, invoking this function places the tape in motion in a reverse direction at fast speed. The transport must be in remote, and the tape must be loaded (see function FM STOP for more detail). The fast speed of the transport is controlled internal to the transport.

[FM REWIND]-----|

WARNING

WARNING

Any tape motion constitutes a hazard to the operator. This motion can be more of a surprise if the commands are remote from the transport.

Assure that the transport door is closed before any tape motion commands are invoked.

Assure that personnel contact with tape reels cannot be accomplished when tape motion can occur.

Take the transport out of remote when loading or unloading tape by pressing any tape speed button on the transport control panel, releasing the remote button. Note that the remote light will remain on to indicate that a remote cable is connected to the transport.

Never engage remote control (never press the remote button) unless the transport door is closed, separating the operator from the tape, and tape reels.

FM SETUP

Function Name: FM SETUP
Function Number: 4310
Module: MODUSR4

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[FM SETUP]-----|

FM STOP

Function Name: FM STOP
Function Number: 4303
Module: MODUSR4

One of the MOTION category of FM tape functions, invoking this function can have two separate effects:

If the tape has not been loaded into the vacuum chambers, tensioning the tape, this function loads the tape.

If the tape has been loaded, this function stops a tape in motion. The transport must be in remote, and the tape must be loaded (see above). The stop time is saved in an internal buffer, and will be logged by function LOG DATA POINT.

[FM STOP]-----|

WARNING

WARNING

Any tape motion constitutes a hazard to the operator. This motion can be more of a surprise if the commands are remote from the transport.

Assure that the transport door is closed before any tape motion commands are invoked.

Assure that personnel contact with tape reels cannot be accomplished when tape motion can occur.

Take the transport out of remote when loading or unloading tape by pressing any tape speed button on the transport control panel, releasing the remote button. Note that the remote light will remain on to indicate that a remote cable is connected to the transport.

Never engage remote control (never press the remote button) unless the transport door is closed, separating the operator from the tape, and tape reels.

FM TAPE

Function Name: FM TAPE
Function Number: 4300
Module: MODUSR4

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[FM TAPE]-----|

FM TAPE SPEED

Function Name: FM TAPE SPEED
Function Number: 4306
Module: MODUSR4

When tape motion functions FM PLAYBACK, FM RECORD, FM AUTOPLAYBACK, FM AUTORECORD, FM FORWARD, and FM REVERSE are invoked, they move tape at the speed set by this function.

```
[ FM TAPE SPEED ]-----|
                  |
                  |--[ = value ]-----|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
value numeric	any valid tape speed	
7.5, 15, 30, 60, or 120		1.875, 3.75,

FM TCG TIME

Function Name: FM TCG TIME
Function Number: 4307
Module: MODUSR4

This function returns the current time according to the time code generator. This function is used to determine the start and stop times during recording functions, and to determine the current tape time during playback functions. The time returned is a real number representing the number of seconds since the beginning of the year. Day of year information is not encoded in this time.

This function is used internally by various functions, and is rarely invoked by the operator.

[FM TCG TIME]-----|

FROM DISK

Function Name: FROM DISK
Function Number: 4026
Module: MODUSR1

This function defines the mass storage device from which the files will be copied - function COPY FILES - or moved - function MOVE FILES.

```
[ FROM DISK ]-----|  
                |  
                |--[ - ]--[ MSI]---
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
value	MSI	any valid HP storage device

GET COEF

Function Name: GET COEF
Function Number: 4122
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[GET COEF]-----|

GET MEAN DATA

Function Name: GET MEAN DATA
Function Number: 4115
Module: MODUSR2

This function retrieves mean data from the low speed data acquisition mechanism, which is typically a data acquisition computer linked via HP-IB to DDAPS. Function SAMPLES TO AVG must have previously defined the number of packets to retrieve and average. The data is averaged and placed in temporary buffers. If function DIGITIZE ENABLE has been set, then mean quantities of pre-programmed digitized traces are buffered. FM tape start/stop times, and Precision Filter gain settings are also buffered. The buffered data is normally logged by function LOG DATA POINT immediately after this function completes execution.

[GET MEAN DATA]----->

GET PF GAINS

Function Name: GET PF GAINS
Function Number: 4140
Module: MODUSR2

This function retrieves gains from the amplifier/filter system for all channels associated with the three wires of each probe in the range of INITIAL PROBE to ENDING PROBE.

The association of wires to channels is assumed to be 1-to-1. The aggregate gain for each wire (channel) is stored in an internal buffer called *Pf_gain*. This buffer is available to function LOG DATA POINT, where the gains can be transferred to the logfile. The first three channels are retrieved to support the first probe, the second three for the second probe, etc.

If the analog cabling is other than this, the arrangement of data in the *Pf_gain* array may map differently. For instance, if two three wire probes are utilized, and if the second three, and fourth three channels are configured to process mean data, instead of fluctuating data, then channels 4, 5, 6, 10, 11, and 12 would relate to DC components of the wire output, and not the fluctuating, more gained, components of the output. This arrangement of the *Pf_gains* array would require modification in the code associated with function LOG DATA POINT.

[GET PF GAINS]-----|

HOTWIRE MENU

Function Name: HOTWIRE MENU
Function Number: 4100
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[HOTWIRE MENU]-----|

HOTWIRE CALC

Function Name: HOTWIRE CALC
Function Number: 4100
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[HOTWIRE CALC]-----|

IDENT

Function Name: IDENT
Function Number: 4150
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[IDENT]-----|

INITIAL OBS

Function Name: INITIAL OBS
Function Number: 4112
Module: MODUSR2

This function declares the initial observation to be processed by other MODUSR2 functions. Functions utilizing this feature to define a range of observations to be processed include START TIME, END TIME, and SET FILENAMES.

```
[ INITIAL OBS ]-----|
                  |
                  |--[ - ]--[value]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
value	numeric integer	1 to ENDING OBS

INITIAL PROBE

Function Name: INITIAL PROBE
Function Number: 4130
Module: MODUSR2

This function declares the beginning probe to be processed by other MODUSR2 functions - see function ENDING PROBE. Functions utilizing this feature to define the range of probes to be processed are: COMPUTE SENS, LOG DATA POINT, CODE TO GAINS, CALC VEL etc, COMPUTE R etc, Remake Probe and LOGFILE TO PC.

```
[ INITIAL PROBE ]-----|  
                    |  
                    |--[ - ]--[value]--|
```

Item	Description	Range
value	numeric integer	1 to 4, the max number of probes

LOAD COEFS

Function Name: LOAD COEFS
Function Number: 4102
Module: MODUSR2

This function loads hotwire calibration coefficients into memory from disk. These coefficients were previously stored by function STORE COEFS.

The coefficient filename must have been previously defined - see function COEF FILENAME.

[LOAD COEFS]-----|

LOAD LOGFILE

Function Name: LOAD LOGFILE
Function Number: 4106
Module: MODUSR2

This function loads the files used to log hotwire calibration observation data. The function LOG FILENAME declares the file name, and function DISC DEVICE declares the Mass Storage Identifier. See LOG FILENAME for a description of the files loaded into memory.

If the logfile does not already exist on the directory and drive defined by DATA DISC DEV, then a new logfile is created. The user is asked to input both a title and a test number. The title is later used to document the file, printouts, and plots. The test number is used by function MARK GOOD DATA to reinstate the TEST variable. (MARK BAD DATA, MARK IGNORE, MARK CAL RAND, MARK CAL SINE, and MARK CAL ZERO change the TEST variable to a coded integer value, and MARK GOOD DATA restores the TEST variable to the test number input by the operator when the logfile is created.

[LOAD LOGFILE]-----|

NOTE:

This function must be performed before the function LOG DATA POINT can be invoked.

LOG DATA

Function Name: LOG DATA
Function Number: 4109
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[LOG DATA]-----|

LOG DATA POINT

Function Name: LOG DATA POINT
Function Number: 4108
Module: MODUSR2

This function logs the hotwire calibration observation data into the logfile(s). The function LOG FILENAME declares the file name, and function DATA DISC DEV declares the Mass Storage Identifier. See LOG FILENAME for a description of the files which receive the data.

This function will not log data that has not been collected and buffered by function GET MEAN DATA. Therefore, GET MEAN DATA must precede each invocation of this function. Functions FM RECORD, FM AUTORECORD, NEFF GAINS, and GET PF GAINS produce internal buffers of data that are integrated into the logfile by this function. These buffers are then marked as invalid. Unless the buffers are updated, the next invocation of this function will not again provide valid buffered data to the next observation in the logfile.

This function also calculates various data items for inclusion in the logfile.

[LOG DATA POINT]-----|

LOG FILENAME

Function Name: LOG FILENAME
Function Number: 4105
Module: MODUSR2

This function selects the name of the file which contains the log data. Function LOAD LOGFILE uses the file name defined here to retrieve (or create) the log data file. Internally, the file name is also used to define the file where log data is stored.

[LOG FILENAME]-----|

LOGFILE TO PC

Function Name: LOGFILE TO PC
Function Number: 4127
Module: MODUSR2

This function transfers logfile data to another computer via HP-IB. The information transmitted is all in ASCII to assure compatibility between systems.

For each probe in the range INITIAL PROBE to ENDING PROBE and for observations in the range INITIAL OBS to ENDING OBS, the appropriate observations records are transmitted. Prior to transmitting the set of each probe's observations, the names of the variables contained in the observation record are transmitted.

[LOGFILE TO PC]-----|

NOTE

Appropriate receiving software is assumed to be in place in the receiving PC. Refer to the appendix for a listing of an appropriate PC BASIC program "XFR.HP".

MARK BAD DATA

Function Name: MARK BAD DATA
Function Number: 4164
Module: MODUSR2

This function identifies observations which, for a variety of reasons have bad data in the observation. The integrity of the data is therefore suspect, and the observation is marked to assist in further data reduction efforts. The TEST NUMBER variable of the current observation, defined by function CURRENT OBS, is changed to become a value of -9999999.99999, which is a "mark" indicating that this observation contains bad data. This "mark" is placed in the "raw" data array, and all "probe" data arrays between the initial probe and the ending probe (see functions INITIAL PROBE and ENDING PROBE), but is not recorded on disk by this function. The "marks" are recorded on disk by function REWRITE LOGFILE.

[MARK BAD DATA]-----|

MARK CAL RAND

Function Name: MARK CAL RAND
Function Number: 4162
Module: MODUSR2

This function identifies observations which are actually calibration observations where the calibration signal is a random noise (typically band limited, for the fluctuating portion of the data system). The TEST NUMBER variable is changed to become a value of -2, which is a "mark" indicating that this observation is in fact a random noise calibration observation. This "mark" is placed in the "raw" data array, and all "probe" data arrays between the initial probe and the ending probe (see functions INITIAL PROBE and ENDING PROBE), but is not recorded on disk by this function. The "marks" are recorded on disk by function REWRITE LOGFILE.

[MARK CAL RAND]-----|

MARK CAL SINE

Function Name: MARK CAL SINE
Function Number: 4161
Module: MODUSR2

This function identifies observations which are actually calibration observations where the calibration signal is a sine wave (for the fluctuating portion of the data system). The TEST NUMBER variable is changed to become a value of -1, which is a "mark" indicating that this observation is in fact a sine wave calibration observation. This "mark" is placed in the "raw" data array, and all "probe" data arrays between the initial probe and the ending probe (see functions INITIAL PROBE and ENDING PROBE), but is not recorded on disk by this function. The "marks" are recorded on disk by function REWRITE LOGFILE.

[MARK CAL SINE]-----|

MARK CAL ZERO

Function Name: MARK CAL ZERO
Function Number: 4163
Module: MODUSR2

This function identifies observations which are actually calibration observations where the calibration signal is zero volts (for the fluctuating portion of the data system). The TEST NUMBER variable is changed to become a value of -3, which is a "mark" indicating that this observation is in fact a zero volts calibration observation. This "mark" is placed in the "raw" data array, and all "probe" data arrays between the initial probe and the ending probe (see functions INITIAL PROBE and ENDING PROBE), but is not recorded on disk by this function. The "marks" are recorded on disk by function REWRITE LOGFILE.

[MARK CAL ZERO]-----|

MARK GOOD DATA

Function Name: MARK GOOD DATA
Function Number: 4166
Module: MODUSR2

This function identifies observations which contain good data, not a calibration observation of any kind, and are not to be ignored in further data reduction. The TEST NUMBER variable is changed to the original test number as first entered when the observation files were first created (see function LOAD LOGFILE). MARK GOOD DATA effectively "erases" any previously set "marks". This "erased mark" is placed in the "raw" data array, and all "probe" data arrays between the initial probe and the ending probe (see functions INITIAL PROBE and ENDING PROBE), but is not recorded on disk by this function. The "erased marks" are recorded on disk by function REWRITE LOGFILE.

[MARK GOOD DATA]-----|

MARK IGNORE

Function Name: MARK IGNORE
Function Number: 4165
Module: MODUSR2

This function identifies observations which are to be ignored in further data reduction. The TEST NUMBER variable is changed to become a value of 0. Normal data reduction of observations with a "mark" (a TEST NUMBER less than, or equal to zero) is inhibited. This "mark" is placed in the "raw" data array, and all "probe" data arrays between the initial probe and the ending probe (see functions INITIAL PROBE and ENDING PROBE), but is not recorded on disk by this function. The "marks" are recorded on disk by function REWRITE LOGFILE.

[MARK IGNORE]-----|

MARK OBS

Function Name: MARK OBS
Function Number: 4160
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[MARK OBS]-----|

MATRIX CHECK

Function Name: MATRIX CHECK
Function Number: 4135
Module: MODUSR2

This function inspects the sensitivity matrix utilized in the CALC VEL etc function. The sensitivities are retrieved for the current probe for each observation as defined by CURRENT PROBE, INITIAL OBS and ENDING OBS. An identity and a determinate are computed, and if the matrix is found to be ill-conditioned, the probe number, observation number, determinate, and accuracy figure are printed. The accuracy figure is 1E-8.

[MATRIX CHECK]-----|

MOVE GROUP

Function Name: MOVE GROUP
Function Number: 4030
Module: MODUSR1

This function defines the files to be selected for moving, and then copy the selected files from a disk device to a disk device using the HP COPY command. When the files are selected, a report is generated on the printer which declares the number of files selected, the from device, the to device, and the files selected for copying. As each file is actually copied, a message is displayed on the CRT, and printed on the printer. Once all selected files have been copied, all successfully copied files are purged from the from device, completing the "move". Functions FROM DISK, TO DISK, AND FILE GROUP all effect the results of this function.

NOTE:

Selection of files by indicating a group name with this function overrides the group name selection previously made by function FILE GROUP.

```
[ MOVE GROUP ]-----|
                    |
                    |--[ - ]--[ group name ]--|
```

Item	Description	Range
group name	string expression	any valid characters that are allowed in a file name see FILE GROUP for details

NEFF

Function Name: NEFF
Function Number: 4400
Module: MODUSR5

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[NEFF]-----|

NEFF GAINS

Function Name: NEFF GAINS
Function Number: 4401
Module: MODUSR5

The gains of the 10 NEFF 130 amplifiers are retrieved as gain codes. These codes are converted to actual gain, and held in a temporary internal buffer called *Neffgain*. If these gains are to be logged, the program code in module MODUSR2 relating to function LOG DATA POINT assigns th gain into variables in the logfile.

[NEFF GAINS]-----|

P1

Function Name: SHOW P1 (P1 is a function name alias)
Function Number: 4191
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the data which is specific to probe 1 (PROBE A).

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[P1]-----|

NOTE:

Function name SHOW P1 cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "P1" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

P2

Function Name: SHOW P2 (P2 is a function name alias)
Function Number: 4192
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the data which is specific to probe 2 (PROBE B).

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[P2]-----|

NOTE:

Function name SHOW P2 cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "P2" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

P3

Function Name: SHOW P3 (P3 is a function name alias)
Function Number: 4193
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the data which is specific to probe 3 (PROBE C).

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[P3]-----|

NOTE:

Function name SHOW P3 cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "P3" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

P4

Function Name: SHOW P4 (P4 is a function name alias)
Function Number: 4194
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the data which is specific to probe 4 (PROBE D).

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[P4]-----|

NOTE:

Function name SHOW P4 cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "P4" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

PEN

Function Name: PEN
Function Number: 4004
Module: MODUSR1

This function sets an internal flag (in MODUSR1) whis is used in various scalar plotting routines. The PEN selection is used to define the color of the plotted parameter. Typical utilization in a sequence program would be to repeatedly invoke the same plot function, but with different data and with a different PEN, without changing the paper in the plotter. In this way, different data sets are plotted in different colors.

[PEN]-----|

PF CONTROL

Function Name: PF CONTROL
Function Number: 4220
Module: MODUSR3

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PF CONTROL]-----|

PF GAINS

Function Name: PF GAINS
Function Number: 4207
Module: MODUSR3

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PF GAINS]-----|

PF INIT

Function Name: PF INIT
Function Number: 4201
Module: MOD7

This function INITIALIZES the Precision Filters, Inc SYSTEM 6000 precision filter/amplifier subsystem. The PF is first reset, precoded module groups are declared within the PF, and default filter and amplifier settings are transferred from the program to the subsystem.

This function is automatically invoked when the program is first executed (Power On), but may need to be manually invoked should the default PF parameters need to be restored to the PF.

[PF INIT]-----|

PF SET PARAMS

Function Name: PF SET PARAMS
Function Number: 4210
Module: MODUSR3

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PF SET PARAMS]-----|

PF STATUS

Function Name: PF STATUS
Function Number: 4206
Module: MODUSR3

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PF STATUS]-----|

PICTURE LOG E

Function Name: PICTURE LOG E
Function Number: 4006
Module: MODUSR1

This function generates x-y plots on the CRT which represent the Log(RhoU) vs. Log(E) for each hotwire. The axes, titles, etc are internally generated using a plot file called "RhoU" - see function PLOT NAME. (It should be noted that this 'RhoU' picture is actually generated for dynamic channels 8 and 9 - which are assumed to be 1 point long, - and offscale as well.)

This function represents data from all observations in the range INITIAL OBS to ENDING OBS for all probes in the range INITIAL PROBE TO ENDING PROBE.

[PICTURE LOG E]-----|

PLAYBACK

Function Name: PLAYBACK
Function Number: 4138
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

It also sets the realtime/playback flag to "P", which is used by function SET FILE NAMES to define the first character of the file name. Function REALTIME changes the flag to "R".

[PLAYBACK]-----|

PLOT EJECT

Function Name: PLOT EJECT
Function Number: 4002
Module: MODUSR1

This function sends a "PG" command to the plot device (if the plot device is not the GRT device).

This function also resets the 'number of observations plotted' pointer, the number of observations printed' pointer, and the 'ending observations' pointer, which affects the operation of functions ENDING OBS, PRNT LOGFILE, PLOT LOG_E and PICTURE LOG_E.

[PLOT EJECT]-----|

PLOT KING
PLOT King

Function Name: PLOT KING
Function Number: 4007,4009
Module: MODUSR1

These two functions execute the same code, and are therefore equivalent.

This function generates x-y plots on the Plotter which represent the Log(U) vs. S(To) for hotwire A. Function PEN defines the plot color. This function is a dynamic function - it can be seen that the current function has little to do with the function name. Typically, the portion of the code (*Plot_king*) in MODUSR1 is dynamically modified to plot according to current data processing requirements.

This function represents data from all observations in the range INITIAL OBS to ENDING OBS for the current probe.

```
[ PLOT KING ]-----|  
[ PLOT King ]-----|
```

PLOT LOG E

Function Name: PLOT LOG E
Function Number: 4005
Module: MODUSR1

This function generates x-y plots on the Plotter which represent the Log(RhoU) vs. Log(E) for each hotwire

This function represents data from all observations in the range INITIAL OBS to ENDING OBS for all probes in the range INITIAL PROBE TO ENDING PROBE.

[PLOT LOG E]-----|

PLOT UTILITYS

Function Name: PLOT UTILITYS
Function Number: 4001
Module: MODUSR1

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PLOT UTILITYS]-----|

PREC FILTERS

Function Name: PREC FILTERS
Function Number: 4200
Module: MODUSR3

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PREC FILTERS]-----|

PRINT DATA

Function Name: PRINT DATA
Function Number: 4117
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PRINT DATA]-----|

PRINT LAST OBS

Function Name: PRINT LAST OBS
Function Number: 4118
Module: MODUSR2

This function prints the data hotwire calibration observation data currently in the logfile(s). The function LOG FILENAME declares the file name, ENDING OBS declares the observation to be printed, and DISC DEVICE declares the Mass Storage Identifier. See LOG FILENAME for a description of the files which contain the data printed. The format of the printout is customized to best demonstrate the hotwire data.

[PRINT LAST OBS]-----|

PRINT LOGFILE

Function Name: PRINT LOGFILE
Function Number: 4107
Module: MODUSR2

This function prints the data hotwire calibration observation data currently in the logfile(s). The function LOG FILENAME declares the file name, and function DATA DISC DEVICE declares the Mass Storage Identifier. See LOG FILENAME for a description of the files which contain the data printed. The format of the printout is customized to best demonstrate the hotwire calibration data. This function prints all observations beginning with INITIAL OBS, and ending with ENDING OBS

[PRINT LOGFILE]-----|

Note: This function was once called PRNT LOGFILE, since the keyword PRINT was reserved in the sequence programming capability of ACQUIRE. That sequence function keyword has been changed from PRINT to SHOW, allowing functions to start with the keyword PRINT.

PRINT SENS

Function Name: PRINT SENS
Function Number: 4119
Module: MODUSR2

This function prints a sensitivity summary of observations currently in the logfile(s), and is intended to demonstrate the velocity, density, and total temperature sensitivity trend of the three wires. The function INITIAL OBS defines the first observation to be summarized, ENDING OBS defines the last observation to be summarized, CURRENT PROBE defines the probe to be summarized, LOG FILENAME declares the logfile which contains the data to be summarized, and DISC DEVICE declares the Mass Storage Identifier of the disk containing the logfile. See LOG FILENAME for a description of the files which contain the data printed.

[PRINT SENS]-----|

PRINT TAPE LOG

Function Name: PRINT TAPE LOG
Function Number: 4116
Module: MODUSR2

This function prints a summary of observations currently in the logfile(s), and is intended to accompany the FM tape reel which had been automatically recorded during a test. Of particular interest in the summary are the variables that specify starting and ending record times of the FM tape for each observation. The function INITIAL OBS defines the first observation to be summarized, ENDING OBS defines the last observation to be summarized, LOG FILENAME declares the file name, and DISC DEVICE declares the Mass Storage Identifier of the disk containing the logfile. See LOG FILENAME for a description of the files which contain the data printed. The format of the printout is customized to best demonstrate the test conditions that existed when the FM tape was recorded beginning with INITIAL OBS, and ending with ENDING OBS.

[PRINT TAPE LOG]-----|

PROBE A

Function Name: PROBE A
Function Number: 4151
Module: MODUSR2

Textual identification of probe A is available for documenting the probe in the log file title, and on printed reports and plots. Text entered by this function becomes a part of the log file title if this function is used before creating a new log file - see LOAD LOGFILE. This function will modify probe identification data used on reports and plots if invoked after LOAD LOGFILE, but will not change the title in the log file.

Text entered is arbitrary, but limited to 10 characters.

[PROBE A]-----|

PROBE B

Function Name: PROBE B
Function Number: 4152
Module: MODUSR2

Textual identification of probe B is available for documenting the probe in the log file title, and on printed reports and plots. Text entered by this function becomes a part of the log file title if this function is used before creating a new log file - see LOAD LOGFILE. This function will modify probe identification data used on reports and plots if invoked after LOAD LOGFILE, but will not change the title in the log file.

Text entered is arbitrary, but limited to 10 characters.

[PROBE B]-----|

PROBE C

Function Name: PROBE C
Function Number: 4153
Module: MODUSR2

Textual identification of probe C is available for documenting the probe in the log file title, and on printed reports and plots. Text entered by this function becomes a part of the log file title if this function is used before creating a new log file - see LOAD LOGFILE.

This function will modify probe identification data used on reports and plots if invoked after LOAD LOGFILE, but will not change the title in the log file.

Text entered is arbitrary, but limited to 10 characters.

[PROBE C]-----|

PROBE D

Function Name: PROBE D
Function Number: 4154
Module: MODUSR2

Textual identification of probe D is available for documenting the probe in the log file title, and on printed reports and plots. Text entered by this function becomes a part of the log file title if this function is used before creating a new log file - see LOAD LOGFILE.

This function will modify probe identification data used on reports and plots if invoked after LOAD LOGFILE, but will not change the title in the log file.

Text entered is arbitrary, but limited to 10 characters.

[PROBE D]-----|

PROCESS UTILITYS

Function Name: PROCESS UTILITYS
Function Number: 4050
Module: MODUSR1

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PROCESS UTILITYS]-----|

PURGE

Function Name: PURGE
Function Number: 4022
Module: MODUSR1

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PURGE]-----|

PURGE FILE

Function Name: PURGE FILE
Function Number: 4024
Module: MODUSR1

This function defines the file to be selected for purging(deleting) from the disk, and then purges the selected file.

```
[ PURGE FILE ]-----|
                    |
                    |--[ - ]--[ name ]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
name	string expression	any valid characters
allowed		that are in a file name

PURGE GROUP

Function Name: PURGE GROUP

Function Number: 4023

Module: MODUSR1

This function purges all selected files. The selection criteria is defined in detail in the HP BASIC language reference manual for HP function PURGE (SELECT).

```
[ PURGE GROUP ]-----|
                        |
                        |--[ - ]--[ name ]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
name	string expression	any valid characters that are allowed in a file name

RAW

Function Name: SHOW RAW (RAW is a function name alias)
Function Number: 4190
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the "raw" data, data which is not probe specific, and data related to general test (tunnel) conditions.

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[RAW]-----|

NOTE:

Function name SHOW RAW cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "RAW" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

READY

Function Name: READY
Function Number: 4175
Module: MODUSR2

This function informs the host computer that DDAPS is ready to take data. A data packet is sent to the host computer (typically the wind tunnel data acquisition computer). This packet contains the "OK" command. The format of the packet is described in 2..

The HPIB bus is configured with the host computer as active controller, and a dedicated HPIB interface card (SC-8) acting as the non active controller (device) on the DDAPS end of the connection.

[READY]-----|

REALTIME

Function Name: REALTIME
Function Number: 4139
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

It also sets the realtime/playback flag to "R", which is used by function SET FILE NAMES to define the first character of the file name. Function PLAYBACK changes the flag to "P".

[REALTIME]-----|

Remake Probe

Function Name: Remake Probe
Function Number: 4128
Module: MODUSR2

This function reproduces the computations normally accomplished during the execution of the function LOG DATA POINT, but without actually acquiring any new data. The purpose is to recompute data should modifications to the computations become necessary.

This function performs these calculations for all probes in the range INITIAL PROBE to ENDING PROBE, and for all observations in the range INITIAL OBS to ENDING OBS.

NOTE:

This function permanently overwrites previous data in the logfiles.

[Remake Probe]-----|

Reorder Obs

Function Name: Reorder Obs
Function Number: 4171
Module: MODUSR2

This function reorders observations. This function exists to manage the logfiles. The possibility exists that data may be taken in an order inappropriate for data reduction, where data blocking with INITIAL OBS and ENDING OBS may be difficult because of the data taking order.

This function rearranges observations for all probes and all observations according to a DATA statement in SUB *Change order*. This data statement defines the picking order when generating the new order of the log file. The *Order_list* is currently configured for a 1-to-1 reordering, and would generate a new order equal to the old order. To generate the desired reordering, it is necessary to modify the *Order_list* before executing this function.

NOTE:

This function permanently overwrites previous data in the logfiles.

[Reorder Obs]-----|

REWRITE LOGFILE

Function Name: REWRITE LOGFILE
Function Number: 4170
Module: MODUSR2

This function transfers all observation records, both "raw" and "probe" (between initial probe and ending probe) to the logfile on disk defined by function LOG FILENAME. For most functions that modify observation data, this function is intrinsic to those functions. However, some functions do not transfer data to disk, and therefore this function must be invoked to generate the disk copy of the observations previously modified only in memory. These functions should be followed by an explicit invocation of REWRITE LOGFILE to assure that the observations are permanently stored on disk. These functions include: MARK CAL SINE, MARK CAL RAND, MARK CAL ZERO, MARK BAD DATA, MARK IGNORE, and MARK GOOD DATA.

[REWRITE LOGFILE]-----|

SAMPLES TO AVG

Function Name: SAMPLES TO AVG
Function Number: 4110
Module: MODUSR2

This function declares the number of data samples to be included in an average of the hotwire calibration data. See LOG DATA POINT for a description of the actual data acquisition process.

[SAMPLES TO AVG]-----|

SCALAR PLOTS

Function Name: SCALAR PLOTS
Function Number: 4019
Module: MODUSR1

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[SCALAR PLOTS]-----|

SELECTOR

Function Name: SELECTOR
Function Number: 4129
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[SELECTOR]-----|

SEND DATA

Function Name: SEND DATA
Function Number: 4500
Module: MODUSR6

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[SEND DATA]-----|

SEND END

Function Name: SEND END
Function Number: 4502
Module: MODUSR6

A data packet is sent to the Spectral Data System, which contains some relevant data. The data includes time of day, the FM tape speed, NEFF 130 gains (see function NEFF GAINS), and the last available start and stop time of the FM tape.

[SEND END]-----|

SEND START

Function Name: SEND START
Function Number: 4501
Module: MODUSR6

A data packet is sent to the Spectral Data System, which contains a "TAKE DATA" message and some relevant data. The data includes time of day, the FM tape speed, NEFF 130 gains (see function NEFF GAINS), and the last available start and stop time of the FM tape.

[SEND START]-----|

SENS ALG

Function Name: SENS ALG

Function Number: 41XX

Module: MODUSR2

This function selects the algorithm to be used when function COMPUTE SENS computes hotwire sensitivities.

These algorithms are:

- 1 - based on $E = f(U, \rho, T_o)$ LOG
- 2 - based on $Nu = f(M, Kn, r)$ LOG (Baldwin)
- 3 - based on $Nu = f(M, Re, \theta)$ LOG (Morkovin)
- 4 - based on $E = f(U, \rho, T_o)$ POLY
- 5 - based on $Nu = f(M, Kn, r)$ POLY (Baldwin)
- 6 - based on $Nu = f(M, Re, \theta)$ POLY (Morkovin)

NOTE:

Function COMPUTE SENS executes only one algorithm each time it is invoked.

```
[ SENS ALG ]-----|
                |--[ algorithm # ]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
-------------	--------------------	--------------

1	algorithm #	numeric integer 1 to 6
---	-------------	--------------------------

SENS KING

Function Name: SENS KING

Function Number: 4141

Module: MODUSR2

This function computes velocity sensitivity (S_u) based on Kings Law; ie.

$$E^2 = A + BU^n$$

Generally, 0.5 is an accepted value of n , and is used in this function. In computing S_u , a least square technique is applied in fitting a curve from INITIAL OBS to ENDING OBS for all three wires of the CURRENT PROBE. Extraneous data values are excluded from the curve fitting.

S_u is not stored in the logfile, but is printed out in a report.

NOTE:

A simple modification in the area of *Sens_king* can convert this function so that it can also compute the mass flow sensitivity (S_m) according to Kings Law.

[SENS KING]-----|

SEQ REMOTE GO

Function Name: SEQ REMOTE GO

Function Number: 4051

Module: MODUSR1

This function is intended for use in a sequence program. Its purpose is to stall the sequence until a remote command is received as the first word in a data packet from the host data acquisition computer. If the remote command is "TAKE DATA", the sequence is released to continue. If the remote command is "LOCAL CONTROL", the sequence program is summarily terminated.

[SEQ REMOTE GO]-----|

SET F FILE NAMES

Function Name: SET F FILE NAMES
Function Number: 41XX
Module: MODUSR2

This function generates filenames based on the CURRENT OBS data in the logfile. These filenames are the same names that are related to the FILENAME function, and can therefore be overwritten by either function, and displayed by the FILENAME function.

Generation of fluctuating computed file names according to a convention allows the automatic processing of data where the relationship of the data to the log file information is known. Typically, this function is invoked to either store computed data for a specific observation, or to retrieve specific fluctuating velocity, density, and total temperature files previously named by this function. The function itself merely defines the filename. Related functions LOAD DATA and STORE DATA use the file names, noting that the disk files are located in the subdirectory and disk defined by function DATA DISC DEV. The disk file names are generated with the format:

character			
1	V	velocity	
	D	density	
	T	temperature	
2	A	probe 1	
	B	probe 2	
	C	probe 3	
	D	probe 4	
3,4	00-99	Run number	
	(2 least significant digits)		
5,6	00-99	Point number	
	(2 least significant digits)		
7,8	—	two underscores	
9,10	unused		
or	00-99	sequence number added if necessary to prevent file overwrite	

Example: "VA0317__"
V -Velocity
A -probe A
03-RUN
17-POINT

[SET F FILE NAMES]-----|

SET FILE NAMES

Function Name: SET FILE NAMES
Function Number: 4180
Module: MODUSR2

This function generates filenames based on the CURRENT OBS data in the logfile. These filenames are the same names that are related to the FILENAME function, and can therefore be overwritten by either function, and displayed by the FILENAME function.

Generation of fluctuating voltage file names according to a convention allows the automatic processing of data where the relationship of the data to the log file information is known. Typically, this function is invoked to either store digitized data for a specific observation, or to retrieve specific fluctuating voltage files previously named by this function. The function itself merely defines the filename. Related functions LOAD DATA and STORE DATA use the file names, noting that The disk files are located in the subdirectory and disk defined by function DATA DISC DEV. The disk file names are generated with the format:

character			
1	R	realtime	set by the REALTIME function
	P	playback	set by the PLAYBACK function
2	A	probe 1	
	B	probe 2	
	C	probe 3	
	D	probe 4	
3,4	00-99	Run number	
	(2 least significant digits)		
5,6	00-99	Point number	
	(2 least significant digits)		
7,8	00-99	memory number	
9,10	unused		
or	00-99	sequence number added if necessary to prevent file overwrite	

Example: "RA031701"
R -Realtime digitization
A -probe A
03-RUN
17-POINT
01-memory 1

[SET FILE NAMES]-----|

SET TIME

Function Name: SET TIME

Function Number: 4050

Module: MODUSR1

This function allows the operator to set the system time and date. The operator is prompted to enter the correct time and date, or to accept the current time and/or date.

Note: This functional responsibility had been available only when the system was first started. This functional area was removed from the baseline ACQUIRE software, and relocated in MODUSR1.

[SET TIME]-----|

SHOW DATA

Function Name: SHOW DATA
Function Number: 4189
Module: MODUSR2

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[SHOW DATA]-----|

SHOW P1

Function Name: SHOW P1
Function Number: 4191
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the data which is specific to probe 1 (PROBE A).

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[SHOW P1]-----|

NOTE:

Function name SHOW P1 cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "P1" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

SHOW P2

Function Name: SHOW P2
Function Number: 4192
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the data which is specific to probe 2 (PROBE B).

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[SHOW P2]-----|

NOTE:

Function name SHOW P2 cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "P2" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

SHOW P3

Function Name: SHOW P3
Function Number: 4193
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the data which is specific to probe 3 (PROBE C).

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[SHOW P3]-----|

NOTE:

Function name SHOW P3 cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "P3" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

SHOW P4

Function Name: SHOW P4
Function Number: 4194
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the data which is specific to probe 4 (PROBE D).

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[SHOW P4]-----|

NOTE:

Function name SHOW P4 cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "P4" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

SHOW RAW

Function Name: SHOW RAW
Function Number: 4190
Module: MODUSR2

Data display of a single observation of data is accomplished on the CRT screen. All variables for one observation are displayed, and may therefore scroll the screen. Use of the arrow cursor keys will allow detailed inspection of all variables.

This function displays the "raw" data, data which is not probe specific, and data related to general test (tunnel) conditions.

Functions LOG FILENAME and DATA DISC DEV define the log file where the data resides. Function LOAD LOGFILE loads the data into the computer where this function accesses it. Function CURRENT OBS defines which observation will be displayed.

WARNING

This function pauses the program for leisurely observation of the displayed data, and requires the operator to continue the program.

[SHOW RAW]-----|

NOTE:

Function name SHOW RAW cannot be used in a sequence program due to a naming conflict with sequence command "SHOW". Function alias "RAW" can be used in a sequence program, but it is not recommended, since this function pauses the program to allow the operator to scroll the screen to view all the variables at leisure.

SIMULT EQ

Function Name: SIMULT EQ
Function Number: 1591
Module: MOD7

A simultaneous equation is solved when this function is invoked. This function is not normally invoked by the operator, since its use is programmatic, in function CALC VEL etc. This function assumes that the inverted sensitivity array has been transferred to the array processor. $\frac{u'}{U}$, $\frac{\rho'}{\rho}$, and $\frac{T_0'}{T_0}$ are returned to trace 1, 2, and 3 respectively.

ALGORITHM

$$\begin{bmatrix} \left[\frac{u'}{U} \right] \\ \left[\frac{\rho'}{\rho} \right] \\ \left[\frac{T_0'}{T_0} \right] \end{bmatrix} = \begin{bmatrix} S_{u_1} & S_{\rho_1} & S_{T_0_1} \\ S_{u_2} & S_{\rho_2} & S_{T_0_2} \\ S_{u_3} & S_{\rho_3} & S_{T_0_3} \end{bmatrix}^{-1} \times \begin{bmatrix} \left[\frac{e'}{E} \frac{1}{G_w} \right]_1 \\ \left[\frac{e'}{E} \frac{1}{G_w} \right]_2 \\ \left[\frac{e'}{E} \frac{1}{G_w} \right]_3 \end{bmatrix}$$

where

$$\begin{aligned} \text{trace 1 contain: } & \left[\frac{e'}{E} \frac{1}{G_w} \right]_1 \\ \text{trace 2 contain: } & \left[\frac{e'}{E} \frac{1}{G_w} \right]_2 \\ \text{trace 3 contain: } & \left[\frac{e'}{E} \frac{1}{G_w} \right]_3 \end{aligned}$$

$$\left. \begin{matrix} S_{u_1} & S_{\rho_1} & S_{T_0_1} \\ S_{u_2} & S_{\rho_2} & S_{T_0_2} \\ S_{u_3} & S_{\rho_3} & S_{T_0_3} \end{matrix} \right\} \text{ assumed to be in the array processor}$$

[SIMULT EQ]-----|

SOUND ALL DONE

Function Name: SOUND ALL DONE
Function Number: 4062
Module: MODUSR1

This function provides an aural indication. It is typically utilized in sequence programs to notify the operator of a completed series functions. A series of rapid tones, each with decreasing frequency, the sound may be familiar to "PACMAN" players: PACMAN being gobbled up.

[SOUND ALL DONE]-----|

SOUND GOT IT

Function Name: SOUND GOT IT
Function Number: 4061
Module: MODUSR1

This function provides an aural indication. It is typically utilized in sequence programs to notify the operator of a completed function - like data acquisition. A series of rapid tones, each with increasing frequency, the sound may be familiar to "PACMAN" players: PACMAN gobbling up the cherry.

[SOUND GOT IT]-----|

START TIME

Function Name: START TIME
Function Number: 4136
Module: MODUSR2

This function returns the FM tape start time for the current observation - according to the log file - and also retrieves search limit times based on the initial observation start time and the ending observation stop time. This function is used to determine the start time during playback functions. The time returned is a real number representing the number of seconds since the beginning of the year. Day of year information is not encoded in this time.

This function is used internally by various functions, and is rarely invoked by the operator.

[START TIME]-----|

START TIME FOUND

Function Name: START TIME FOUND
Function Number: 4331
Module: MODUSR4

When invoking function FM AUTOPLAYBACK or function FM POSITION, a search for a start time is invoked. When the time is found, an internal flag is set. This function returns the value of that flag.

If 0 is returned, the start time was not found. This can occur if the stop time does not exist on the tape, or if the start time is outside the search limits defined by function FMPOSITION.

If 1 is returned, The start time was found.

Use of this function in a sequence program can control alternate processes if the time could not be found.

[START TIME FOUND]-----|

STOP TIME FOUND

Function Name: STOP TIME FOUND
Function Number: 4332
Module: MODUSR4

When invoking function FM AUTOPLAYBACK or function FM POSITION, a search for a stop time is invoked. When the time is found, an internal flag is set. This function returns the value of that flag.

If 0 is returned, the stop time was not found. This can occur if the stop time does not exist on the tape, or if the stop time is outside the search limits defined by function FMPOSITION.

If 1 is returned, The stop time was found.

Use of this function in a sequence program can control alternate processes if the time could not be found.

[STOP TIME FOUND]-----|

STORE C DATA ASC

Function Name: STORE C DATA ASC
Function Number: 268
Module: MODGEN

This function stores the region of fluctuating data between the cursors in ASCII format from the memories defined by function DATA FILE MAP into files defined by function FILENAME in a subdirectory called ASCII under a directory and disk defined by function DATA DISC DEV.

This function is very similar to STORE C DATA, but the data is converted to real ASCII format before it is sent to disk. The files are much larger than the binary data files generated by STORE C DATA. The binary representation of a single sample takes 2 bytes, where the ASCII representation takes about 20 bytes (characters).

Implementation of this function was accomplished by modifying MODGEN, a program module supplied as a part of ACQUIRE by Data Laboratories, Ltd.

NOTE:

For 256K samples, over 5Mb of disk space is required.

For 256K samples, more than 20 minutes will be required to translate the binary data (in the memories) to real ASCII, and store the strings, one sample per line, on disk.

[STORE C DATA ASC]-----|

STORE COEFS

Function Name: STORE COEFS
Function Number: 4124
Module: MODUSR2

This function stores coefficients previously defined by functions ENTER COEFS, LOAD COEFS, COMPUTE COEFS, etc.

The coefficient filename must have been previously defined - see function COEF FILENAME.

The data disk device must have been previously defined - see function DATA DISC DEV.

[STORE COEFS]-----|

STORE DATA ASC

Function Name: STORE DATA ASC
Function Number: 269
Module: MODGEN

This function stores fluctuating data in ASCII format from the memories defined by function DATA FILE MAP into files defined by function FILENAME in a subdirectory called ASCII under a directory and disk defined by function DATA DISC DEV.

This function is very similar to STORE DATA, but the data is converted to real ASCII format before it is sent to disk. The process is a lengthy one, and the files are much larger than the binary data files generated by STORE DATA. The binary representation of a single sample takes 2 bytes, where the ASCII representation takes about 20 bytes (characters).

Implementation of this function was accomplished by modifying MODGEN, a program module supplied as a part of ACQUIRE by Data Laboratories, Ltd.

NOTE:

For 256K samples, over 5Mb of disk space is required.

For 256K samples, more than 20 minutes will be required to translate the binary data (in the memories) to real ASCII, and store the strings, one sample per line, on disk.

[STORE DATA ASC]-----|

TAG PICTURE

Function Name: TAG PICTURE
Function Number: 4004
Module: MODUSR1

This function "tags" the picture with the first few parameter names and values from the current observation. These few values are intended to identify the environment from which the "picture" was taken. This function is therefore intended to be invoked just after function REDRAW PICTURE or PICTURE LOG E.

[TAG PICTURE]-----|

TAG PLOT

Function Name: TAG PLOT
Function Number: 4003
Module: MODUSR1

This function "tags" the plot with the first few parameter names and values from the current observation. These few values are intended to identify the environment from which the plot was taken. This function is therefore intended to be invoked just after function REDRAW PLOT or PLOT LOG E.

[TAG PLOT]-----|

TO DISK

Function Name: TO DISK
Function Number: 4027
Module: MODUSR1

This function defines the mass storage device to which the files will be copied - function COPY FILES - or moved - function MOVE FILES.

```
[ TO DISK ]-----|
                |
                |--[ - ]--[ MSI ]--|
```

<u>Item</u>	<u>Description</u>	<u>Range</u>
value	MSI	any valid HP storage device

UTILITY

Function Name: UTILITY
Function Number: 4000
Module: MODUSR1

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[UTILITY]-----|

< blank page >

APPENDIX B. Processing Comparison Results

< blank page >

DDAPS CHECKOUT

The development of the Dynamic Data Acquisition System (DDAPS) requires periodic checking of the algorithms implemented within. In May, 1990, a check of the ability of DDAPS to process TRU three-wire anemometer data was accomplished after the addition of calibration mechanisms and the attachment of an array processor to speed up the processing of vectors that are 256K samples long.

Comparison was made to results of an independent data reduction algorithm performed via ACD by Mr. Glenn Bittner (Unisys). Although all mean quantities (local conditions, wire mean voltages, amplifier gains and wire sensitivities) were generated by DDAPS, all fluctuating data was processed independently from the same FM tape. Digitization rates and bandwidth were the same for both reductions - 12,500 samples/second, 1Hz to 5KHz.

Observation 29 through 36 represent one mach sweep at atmosphere (120°F) at the LTPT. Observation 32 is a bad data point (the anemometers were in standby). This data is from test 346, where various three wire probe configurations were compared. This data is from a three wire probe with wires in a Y configuration.

Note:

When observation 29 was digitized, the time trace suggested that a particle impacted wire 1 within the 20 second record, which seemed to contribute to both the rms voltages and to the resultant turbulence measurement. Observation 29, was re-digitized by DDAPS, utilizing a slightly different segment of the 30 second recording.

Variations in results between the results of the two implementations are probably attributable to the arbitrary selection of the 20 seconds (of the 30 seconds recorded on FM tape) digitized.

PRECEDING PAGE BLANK NOT FILMED

PAGE 244 INTENTIONALLY BLANK

120⁰ F; 1ATM @LTPT

		e(rms) ₁		e(rms) ₂		e(rms) ₃	
		ACD	DDAPS	ACD	DDAPS	ACD	DDAPS
		mvolts	mvolts	mvolts	mvolts	mvolts	mvolts
OBS	M _∞	-----	-----	-----	-----	-----	-----
29	.37	4.474	4.503	1.802	1.854	3.319	3.772
29*			4.423		1.78		3.149
30	.3	3.025	2.985	1.241	1.235	1.945	1.918
31	.25	2.752	2.726	1.061	1.065	1.501	1.489
32		Bad data point (anemometers in standby)					
33	.2	2.058	2.085	0.832	0.832	1.187	1.159
34	.15	1.347	1.309	0.599	0.604	0.956	0.979
35	.1	1.185	1.189	0.6089	0.619	0.9968	1.001
36	.05	1.230	1.03	0.6549	0.568	1.088	0.921

		u'/U(rms)		ρ'/ρ(rms)		T'o/T'o(rms)	
		ACD	DDAPS	ACD	DDAPS	ACD	DDAPS
		%	%	%	%	%	%
OBS	M _∞	-----	-----	-----	-----	-----	-----
29	.37	0.933	1.408	0.623	0.950	0.0251	0.0347
29*			0.779		0.522		0.0221
30	.3	0.771	0.760	0.501	0.493	0.0226	0.0223
31	.25	0.871	0.871	0.581	0.580	0.0200	0.0199
32		Bad data point (anemometers in standby)					
33	.2	0.763	0.777	0.516	0.524	0.0176	0.0178
34	.15	0.578	0.564	0.399	0.391	0.0150	0.0148
35	.1	0.397	0.444	0.264	0.276	0.0145	0.0150
36	.05	2.154	2.348	1.523	1.659	0.0825	0.0892

APPENDIX C. Sequence Programs

< blank page >

AUTOSEQ

1 DISC DEVICE-/TEST:,1406
2 DATA DISC DEV-/TEST/DATA:,1406
10 SEQ PROG NAME-S_INIT
20 LOAD SEQ PROG

PREVIOUS PAGE BLANK NOT FILMED

~~248~~ 248 INTENTIONALLY BLANK

S_INIT

```
1  GRAPH OFF
12 MEM LENGTH-256K
13 MEM LENGTH(4)-4096
14 MEM LENGTH(5)-4096
15 MEM LENGTH(6)-4096
16 MEM START-256K
17 MEM START(5)-772K
18 MEM START(6)-776K
19 MEM LENGTH(6)-4096
35 NUM MEMS-6
36 CHAN IN USE-YES
40 BIN SW NAME-SWTEST
50 CHAN IN USE-YES
60 LOAD BIN SW
63 TRACE ACTIVE-1,2,3,4,5,6
64 TRACE ON-1,2,3,4,5,6
65 TR CRT XMIN(4)-96
66 TR CRT XMIN(5)-96
67 TR CRT XMIN(6)-96
68 TR CRT XMAX(1)-95
69 TR CRT XMAX(2)-95
70 TR CRT XMAX(3)-95
76 TR CRT YMAX(4)-99
77 TR CRT YMIN(4)-67
78 TR CRT YMAX(5)-66
79 TR CRT YMIN(5)-34
80 TR CRT YMAX(6)-33
81 TR CRT YMIN(6)-0
82 TR CRT YMAX(1)-99
83 TR CRT YMIN(1)-67
84 TR CRT YMAX(2)-66
85 TR CRT YMIN(2)-34
86 TR CRT YMAX(3)-33
87 TR CRT YMIN(3)-0
88 TR CRT YMAX(3)-33
90 FORMAT NUMBER-8
91 SAVE FORMAT
92 XEXPAND-1
110 TF MAP CHAN-1
160 C-ALL
170 C TO WHOLE
180 DATA FILE MAP-YES
190 DIGITIZE ENABLE-1
195 SAMPLES TO AVG-1
200 GRAPH ON
300 SEQ PROG NAME-S_CALIB
310 LOAD SEQ PROG
```


S_CALIB

```
50 LOG FILENAME-346
60 LOAD LOGFILE
80 GRAPH OFF
81 TRACE ACTIVE-1,2,3,4,5,6
82 TRACE ON-1,2,3,4,5,6
83 Y CALIB TYPE-VOLTS
84 CHAN IN USE=YES
85 TF MAP CHAN-1
86 SHOW "Enter 1-FULL CAL, 2-AC only, 3-DC only, 4-TAPE CAL"
87 SHOW "Enter 5-REDUCE (No CAL), 6-PROCESS (No CAL), 9-Nothing"
88 INPUT C
90 IF C=9 THEN STOP SEQ PROG
93 IF C=6 THEN GOTO 970
94 IF C=5 THEN GOTO 950
95 IF C=4 THEN GOTO 900
96 IF C>2 THEN GOTO 298
97 CHAN IN USE(4)=NO
98 CHAN IN USE(5)=NO
99 CHAN IN USE(6)=NO
101 PF PRE GAIN(1)=1
102 PF PRE GAIN(2)=1
103 PF PRE GAIN(3)=1
104 PF PRE GAIN(4)=1
105 PF PRE GAIN(5)=1
106 PF PRE GAIN(6)=1
111 PF LP PRE AMP(1)=1
112 PF LP PRE AMP(2)=1
113 PF LP PRE AMP(3)=1
114 PF LP PRE AMP(4)=1
115 PF LP PRE AMP(5)=1
116 PF LP PRE AMP(6)=1
121 PF POST GAIN(1)=1
122 PF POST GAIN(2)=1
123 PF POST GAIN(3)=1
124 PF POST GAIN(4)=1
125 PF POST GAIN(5)=1
126 PF POST GAIN(6)=1
140 CONST K=.707
141 SHOW "Calibration is @ 90% Full Scale"
142 CONST K=[CONST K]*(.90)
150 SHOW "Set calibrator for [CONST K] Vrms at 67 Hz"
151 WAIT ?
200 WAIT ?
210 ARM
211 TRIGGER
212 WAIT *
213 TF FROM REC
215 CLEAR REC
248 CONST K=1
249 CONST K=[CONST K]*(GET CAL RMS)
251 YCAL1 AT +FS(1)=[CONST K]/[AP RMS(1)]
252 YCAL1 AT +FS(2)=[CONST K]/[AP RMS(2)]
253 YCAL1 AT +FS(3)=[CONST K]/[AP RMS(3)]
260 CONST K=[CONST K]*-1
261 YCAL2 AT -FS(1)=[CONST K]/[AP RMS(1)]
262 YCAL2 AT -FS(2)=[CONST K]/[AP RMS(2)]
263 YCAL2 AT -FS(3)=[CONST K]/[AP RMS(3)]
269 CONST K=[CONST K]*-1
270 SHOW " CHAN 1,2,3 ARE CALIBRATED . . . . . "
271 CALIB CHANS
298 IF C=2 THEN GOTO 450
```

```
301 CHAN IN USE(1)-NO
302 CHAN IN USE(2)-NO
303 CHAN IN USE(3)-NO
304 CHAN IN USE(4)-YES
305 CHAN IN USE(5)-YES
306 CHAN IN USE(6)-YES
320 CONST K=10
321 SHOW "Calibration is @ 90% Full Scale"
322 CONST K=[CONST K]*(.90)
323 SHOW "SET CALIBRATOR FOR [CONST K] V DC"
325 WAIT ?
340 ARM
341 TRIGGER
342 WAIT *
343 TF FROM REC
344 CLEAR REC
347 CONST K=10
348 CONST K=[CONST K]*(GET CAL DC)
354 YCAL1 AT +FS(4)=[CONST K]/[MEAN(4)]
355 YCAL1 AT +FS(5)=[CONST K]/[MEAN(5)]
356 YCAL1 AT +FS(6)=[CONST K]/[MEAN(6)]
360 CONST K=[CONST K]*-1
364 YCAL2 AT -FS(4)=[CONST K]/[MEAN(4)]
365 YCAL2 AT -FS(5)=[CONST K]/[MEAN(5)]
366 YCAL2 AT -FS(6)=[CONST K]/[MEAN(6)]
369 CONST K=[CONST K]*-1
370 SHOW "CHAN 4,5,6 ARE CALIBRATED . . . . ."
450 Y CALIB NAME=VOLTSc
460 CHAN IN USE=YES
500 CALIB CHANS
600 Y CALIB TYPE=USER
800 SEQ PROG NAME=S_TAKEDATA
801 LOAD SEQ PROG
900 SEQ PROG NAME=S_CALTAPE
901 LOAD SEQ PROG
950 SEQ PROG NAME=S_REDUCE
951 LOAD SEQ PROG
970 SEQ PROG NAME=S_PROCESS
971 LOAD SEQ PROG
```

S_CALTAPE

```
1  GRAPH OFF
2  LOG FILENAME
3  CURRENT OBS
4  CONST K
10 SHOW "TAPE CALIBRATION RETRIEVAL"
11 SHOW "
12 SHOW "Assure the observation # containing the AC CAL "
13 SHOW "sine wave is set in CURRENT OBS . . ."
14 SHOW " and CONST K is set to the rms level (in volts) . . ."
15 SHOW "IF NOT, press STOP SEQ PROG, then ENTER....."
16 SHOW "IF OK, just press ENTER."
17 SEQUENCE MENU
20 WAIT ?
30 INITIAL OBS=1
31 ENDING OBS=[CURRENT OBS]
32 INITIAL OBS=[CURRENT OBS]
40 PF REMOTE
41 PF PRE GAIN(1)=4
42 PF PRE GAIN(2)=4
43 PF PRE GAIN(3)=4
47 PF HPF(1)=1
48 PF HPF(2)=1
49 PF HPF(3)=1
51 PF LP PRE AMP(1)=1
52 PF LP PRE AMP(2)=1
53 PF LP PRE AMP(3)=1
57 PF LPF(1)=5000
58 PF LPF(2)=5000
59 PF LPF(3)=5000
61 PF POST GAIN(1)=.17
62 PF POST GAIN(2)=.17
63 PF POST GAIN(3)=.17
80 GRAPH OFF
81 TRACE ACTIVE=1,2,3,4,5,6
82 TRACE ON=1,2,3,4,5,6
88 Y CALIB TYPE=VOLTS
90 CHAN IN USE=YES
91 TF MAP CHAN=1
97 CHAN IN USE(4)=NO
98 CHAN IN USE(5)=NO
99 CHAN IN USE(6)=NO
100 BIN SW NAME=SW346
101 LOAD BIN SW
110 PLAYBACK
111 FM STOP
112 FM POSITION
113 LET F=[START TIME FOUND]
114 SHOW F
115 IF F=0 THEN GOTO 900
116 DIGITIZE ENABLE
117 FM AUTOPLAYBACK
118 LET F=[START TIME FOUND]
119 SHOW F
120 IF F=0 THEN GOTO 900
123 ARM
124 TRIGGER
125 WAIT *
126 FM STOP
127 TF FROM REC
201 PSHOW "CALIBRATION MEAN VOLTAGE - WIRE A: [MEAN(1)]"
202 PSHOW "CALIBRATION MEAN VOLTAGE - WIRE B: [MEAN(2)]"
```

```

203 PSHOW "CALIBRATION MEAN VOLTAGE - WIRE C: [MEAN(3)]"
204 PSHOW "CALIBRATION RMS VOLTAGE - WIRE A: [RMS(1)]"
205 PSHOW "CALIBRATION RMS VOLTAGE - WIRE B: [RMS(2)]"
206 PSHOW "CALIBRATION RMS VOLTAGE - WIRE C: [RMS(3)]"
210 GOTO 251
211 AP REMOVE MEAN(1)
212 AP REMOVE MEAN(2)
213 AP REMOVE MEAN(3)
251 YCAL1 AT +FS(1)=[CONST K]{/}[RMS(1)]
252 YCAL1 AT +FS(2)=[CONST K]{/}[RMS(2)]
253 YCAL1 AT +FS(3)=[CONST K]{/}[RMS(3)]
259 CONST K=[CONST K]{*}-1
261 YCAL2 AT -FS(1)=[CONST K]{/}[RMS(1)]
262 YCAL2 AT -FS(2)=[CONST K]{/}[RMS(2)]
263 YCAL2 AT -FS(3)=[CONST K]{/}[RMS(3)]
264 CONST K=[CONST K]{*}-1
270 SHOW " CHAN 1,2,3 ARE CALIBRATED . . . . ."
277 SHOW "
450 Y CALIB NAME-VOLTSc
460 CHAN IN USE-YES
500 CALIB CHANS
600 Y CALIB TYPE-USER
797 SHOW "
798 SHOW " DO N O T change PF gains; they are included in the calibration."
799 SHOW "
800 SEQ PROG NAME-S_PLAYBACK
801 LOAD SEQ PROG
900 SOUND ALL DONE
901 SOUND GOT IT
902 SOUND ALL DONE
903 SHOW "
904 SHOW " CHECK THE OBSERVATION #, AND FOR PROPER FM TAPE . . . ."
905 GO TO 10

```

S_TAKEDATA

```
1  CHAN IN USE=YES
9  SAMPLES TO AVG=1
10 GRAPH OFF
11 C=ALL
12 C TO WHOLE
13 XEXPAND=1
14 BIN SW NAME=SW345
15 LOAD BIN SW
16 SHOW " WAITING FOR TRIGGER FROM HOST"
17 SHOW "-----"
18 READY
20 WAIT ?
21 GRAPH OFF
22 SHOW "Recording takes 30 seconds . . . . ."
23 PF REMOTE
25 ARM
26 TRIGGER
27 WAIT *
32 TF FROM REC
33 AP REMOVE MEAN(1)
35 AP REMOVE MEAN(2)
36 AP REMOVE MEAN(3)
37 GET MEAN DATA
38 WAIT !
39 PF LOCAL
40 SOUND GOT IT
42 GRAPH ON
45 LOG DATA POINT
49 CURRENT OBS=[ENDING OBS|
51 SET FILE NAMES
55 STORE DATA
100 COMPUTE SENS.
150 CALC VEL etc
380 PRINT LAST OBS
390 SOUND ALL DONE
400 GOTO 16
```

S_REDUCE

```
10 SHOW " PRESS ENTER TO TAKE DATA"
19 READY
20 WAIT ?
21 GRAPH OFF
22 SHOW "Recording takes 30 seconds . . . . "
25 ARM
26 TRIGGER
27 WAIT *
29 TF FROM REC
33 AP REMOVE MEAN(1)
34 GOTO 40
35 AP REMOVE MEAN(2)
36 AP REMOVE MEAN(3)
40 GRAPH ON
43 GET MEAN DATA
44 WAIT !
45 LOG DATA POINT
46 SOUND GOT IT
51 SET FILE NAMES
53 DATA FILE MAP(2)-NO
54 DATA FILE MAP(3)-NO
55 STORE DATA
100 COMPUTE SENS.
150 CALC VEL etc
380 PRINT LAST OBS
390 SOUND ALL DONE
400 GO TO 10
```

S_CALC

```
1  CHAN IN USE=YES
2  CHAN IN USE(4)=NO
3  CHAN IN USE(5)=NO
4  CHAN IN USE(6)=NO
60 LET I={INITIAL OBS}
61 LET E={ENDING OBS}
63 LET C={INITIAL OBS}
65 SHOW C
66 SHOW I
67 SHOW E
70 ENDING OBS={C}
71 INITIAL OBS={C}
72 CURRENT OBS={C}
75 INITIAL OBS
76 ENDING OBS
77 CURRENT OBS
80 SHOW " COMPUTING OBSERVATION {CURRENT OBS} . . . "
150 CALC VEL etc
380 PRINT LAST OBS
390 SOUND ALL DONE
391 LET C={C}+1
393 IF C>E THEN GOTO 395
394 GOTO 65
395 SOUND ALL DONE
396 SOUND ALL DONE
397 SOUND ALL DONE
```

S_PROCESS

```
20 GRAPH OFF
25 LOG FILENAME
30 INITIAL OBS
31 ENDING OBS
32 CURRENT OBS
33 PLAYBACK
34 SET FILE NAMES
35 FILENAME
100 SHOW "
101 SHOW "CALCULATING Velocity, Density, & Total Temperature Fluctuations.."
102 SHOW "-----"
110 SEQUENCE MENU
111 SHOW " If the setup is incorrect, STOP SEQ PROG, ENTER."
112 SHOW " If the setup is correct, ENTER."
150 WAIT ?
170 GRAPH ON
200 CALC VEL etc
225 YSHIFT=0
230 FILENAME
240 COPY
300 SOUND GOT IT
500 PRINT LOGFILE
700 SOUND ALL DONE
```


S_PLAYBACK

```
2  SEQUENCE MENU
3  GRAPH OFF
4  SHOW "
5  SHOW "T A P E   D I G I T I Z A T I O N"
6  SHOW "- - - - -"
8  LOG FILENAME
9  CURRENT PROBE
10 PROBE A ID
11 INITIAL OBS
12 ENDING OBS
13 CURRENT OBS
14 SHOW "If NOT OK, then STOP SEQ PROG, ENTER."
15 SHOW "If OK, then ENTER."
16 WAIT ?
17 PLAYBACK
18 FM STOP
19 LET N=0
20 LET I=[INITIAL OBS]
30 LET E=[ENDING OBS]
40 LET C=[CURRENT OBS]
45 CHAN IN USE(1)-YES
46 CHAN IN USE(2)-YES
47 CHAN IN USE(3)-YES
48 CHAN IN USE(4)-NO
49 CHAN IN USE(5)-NO
50 CHAN IN USE(6)-NO
55 TRACE ON=1,2,3
100 FM POSITION
120 LET F=[START TIME FOUND]
121 SHOW F
130 IF F=0 THEN GOTO 500
200 DIGITIZE ENABLE=1
210 GRAPH OFF
211 DATA FILE MAP=NO
212 DATA FILE MAP(1)=YES
213 DATA FILE MAP(2)=YES
214 DATA FILE MAP(3)=YES
215 SET FILE NAMES
220 FILENAME
250 FM AUTOPLAYBACK
260 LET F=[START TIME FOUND]
261 SHOW F
270 IF F=0 THEN GOTO 400
300 SOUND GOT IT
309 CURRENT OBS
310 ARM
320 TRIGGER
330 WAIT *
331 FM STOP
335 TF FROM REC
336 TRACE ON=1,2,3
337 AP REMOVE MEAN(1)
338 AP REMOVE MEAN(2)
339 AP REMOVE MEAN(3)
350 LET F=[STOP TIME FOUND]
351 SHOW F
360 IF F=1 THEN GOTO 400
380 STORE DATA
390 SOUND ALL DONE
395 LET N=[N]+1
400 LET C=[C]+1
```

```
410 IF C>E THEN GOTO 500
420 CURRENT OBS-[C]
490 GOTO 210
500 FM STOP
540 IF N=0 THEN GOTO 900
550 SEQPROG NAME-S_PROCESS
560 LOAD SEQ PROG
900 SEQUENCE MENU
910 STOP SEQ PROG
```

S_PSD

```
2  NUM MEMS=5
5  CURRENT OBS=27
10 DATA FILE MAP(1)=NO
11 DATA FILE MAP(2)=NO
12 DATA FILE MAP(3)=YES
13 DATA FILE MAP(4)=YES
14 DATA FILE MAP(5)=NO
17 FILENAME(3)=VA6225
18 FILENAME(4)=DA6225
20 TRACE ACTIVE=1,2
21 TRACE ON=1,2
100 PLOT NAME =FREQ_TEST
200 MEM START=128K
210 MEM LENGTH=128K
211 LOAD PLOT
215 LET B=512
230 C=ALL
240 C TO WHOLE
250 XEXPAND=1
260 GRAPH ON
300 SELECT MODE=NORMALISE
401 LET L=8
402 LET A=1
410 SHOW "Processing block [A] of size [B] - PS LOG MAG"
411 T(1)=1
412 T(2)=2
420 M(1)=3
421 M(2)=4
425 X CALIB TYPE(1)=ADDR
426 X CALIB TYPE(2)=ADDR
430 C=1,3
431 C TO POSITION=[A](-)1(*)[B]
435 C=2,4
436 C TO POSITION=[A](*)[B](-)1
440 C=ALL
442 XEXPAND(1)=2
443 XEXPAND(2)=2
497 X CALIB TYPE(1)=TIME
498 X CALIB TYPE(2)=TIME
540 PS LOG MAG
545 XEXPAND=2
546 GRAPH ON
547 SOUND GOT IT
548 WAIT 2
550 T(1)=5
551 T(2)=1
553 X CALIB TYPE(2)=ADDR
554 C TO POSITION(3)=0
555 C TO POSITION(4)=[B]()/2
556 C=ALL
557 X CALIB TYPE(2)=TIME
558 XEXPAND=2
559 IF A>1 THEN GOTO 563
561 MEM A:MEM B
562 GOTO 565
563 MEM A+MEM B
565 IF A=L THEN GOTO 700
566 LET A=A+1
575 GOTO 410
700 AP CONST K=1()/[L]
702 T(1)=5
```

```
703 X CALIB TYPE-ADDR
704 C-ALL
705 C TO POSITION(1)-0
706 C TO POSITION(2)-[B](-)1
710 AP K*MEM
740 GRAPH ON
741 STOP SEQ PROG
745 TAG PLOT
750 PLOT PICTURE
```

S_ENSEMBLE

```
100 PLOT NAME -FREQ_TEST
101 LOAD PLOT
200 PRINT "ASSURE DATA STARTS IN MEMORY 1"
201 M(3)-1
210 T-1
300 SELECT MODE-NORMALISE
316 C TO WHOLE
400 PRINT "INPUT BUFFER SIZE"
404 LET S-65535
405 LET B-16383
410 PRINT "PROCESSING 16K BLOCKS"
450 LET I-0
460 LET J-[B]
475 T-1
480 C-3
481 C TO POSITION-0
482 C-4
483 C TO POSITION-2047
485 C-3,4
486 XEXPAND(2)-2
500 M(1)-3
501 X CALIB TYPE(1)-ADDR
502 C-1
503 C TO POSITION -[I]
504 C-2
505 C TO POSITION -[J]
506 C-1,2
510 LP FIL DECI
511 LP FIL DECI
512 LP FIL DECI
540 RFFT LOG MAG
550 REDRAW PICTURE
560 T(2)-1
561 T(1)-2
565 IF I-0 THEN M(2)-1
566 IF I>0 THEN MEM A+MEM B
567 T-1
575 C TO WHOLE
580 LET I-1+[J]
581 LET J-[I]+[B]
599 IF J<[S] THEN GOTO 500
700 CONST K=.0625
710 K*MEM(2)
720 M(1)-2
740 GRAPH ON
750 PLOT PICTURE
```

< blank page >

APPENDIX D. FM Tape and Time Code Generator Interface

PAGE 264 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

May 24, 1988

To: 359/ G. S. Jones
From: 359/ S. J. Clukey
Subject: FM Tape Control Interface

The requirement to reduce a large amount of data previously recorded on FM tape has prompted me to develop a semi automatic data reduction system.

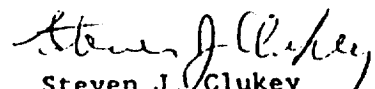
The primary components are the tape transport, the time code generator, the control computer, and the data processing instrument. In operation, the operator would enter the table of times to be processed on the tape, load the appropriate tape, select the channels (2 or 3) to be processed (via a patch panel or manual switch), and enable the processing instrument. When the program is activated, the tape would be positioned to the correct start times, the instrument would be commanded to start processing when the tape was properly positioned and at the proper speed, and the sequence would be repeated for all times in the table.

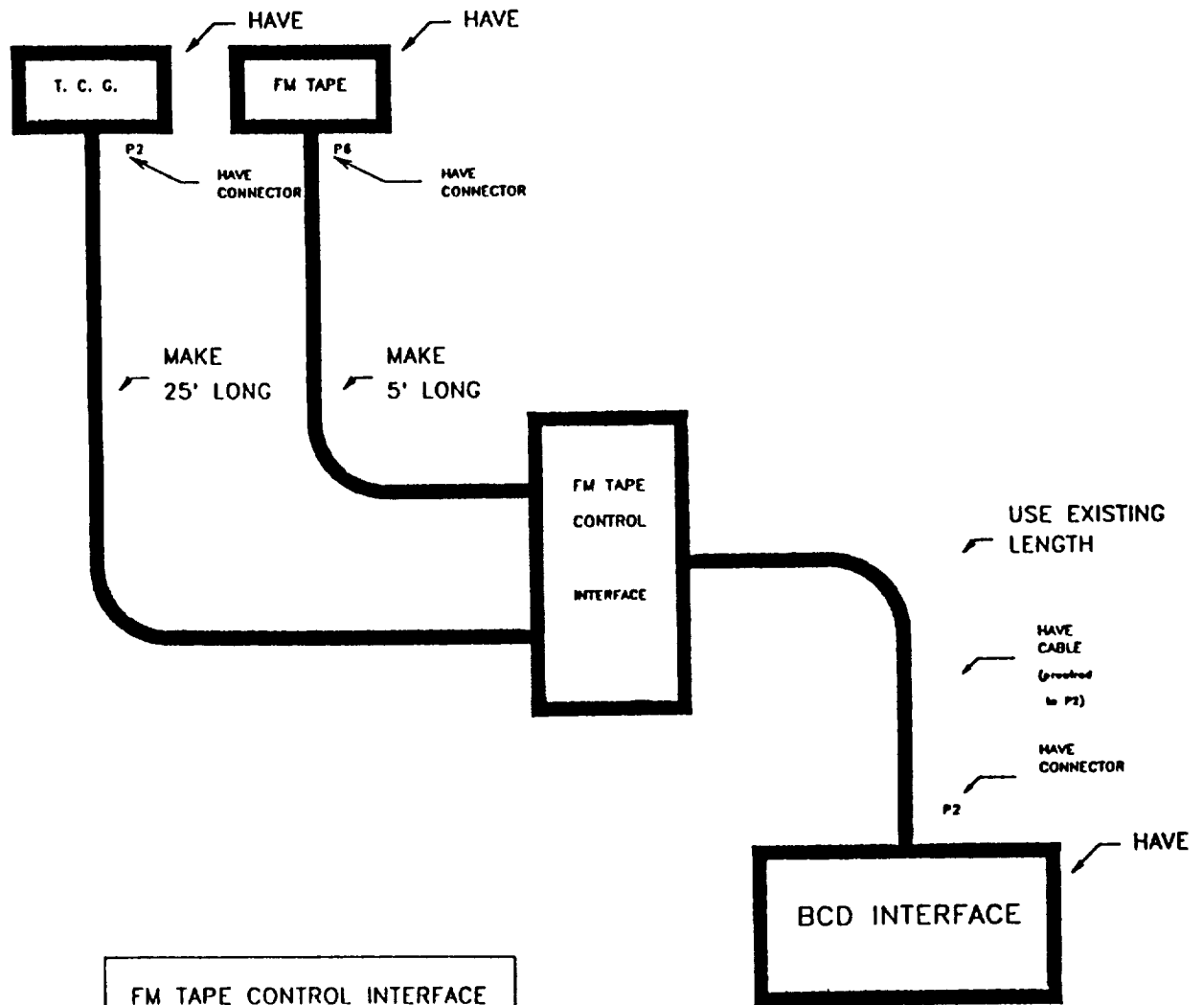
Almost all components are on hand. The programming will be implemented in functional modules, the computer-to-instrument interface is in place, and the interface between the computer, the FM tape, and the time code generator is all that remains to be provided. The attached sketches provide the concept for the interface, and were generated after discussions with Willis Jackson of Wyle Laboratories, who is familiar with the FM tape recorder. Ann Bare (16') provided some financial support by allowing us to use Job Order No. 90095.

I have passed these sketches on to Garry Koeppel, and he will generate the work order to complete the engineering (mechanical, pinouts, etc.) and to fabricate the interface. He recognizes that this interface is to be a prototype of simple construction, and is not to cost a lot of time and money. I suggested to him that the board be a perf board, and the three cables simply be hard-wired to the board (with simple strain relief).

Mr. Greg Chichester of UNISYS has begun the process of providing the software to drive this interface. As the pieces all become available, the actual connection of all the software modules will be implemented in a BASIC program.

I hope that this semi automatic data processing scheme can be available in late August.


Steven J. Clukey
Vigyan Research Associates



FM TAPE CONTROL INTERFACE

CABLE LAYOUT

5/23/68 S. CLURRY VIGYAN RESEARCH ASSOCIATES

BCD
INTERFACE

J1

28 — DO-0 (LSB)
61 — DO-1
60 — DO-2

48 —
49 —
50 —
51 — LOGIC
52 — GROUND

29 — DO-3
28 — DO-4

58 — DO-5
58 — DO-6 (MSB)
27 — DO-7
17 — SGN2

35 — DI1-8
3 — 4
8 — 2
40 — 1

48 — DI2-8
14 — 4
20 — 2
55 — 1

38 — DI3-8
4 — 4
8 — 2
41 — 1

47 — DI4-8
15 — 4
21 — 2
24 — 1

37 — DI5-8
5 — 4
10 — 2
42 — 1

45 — DI6-8
18 — 4
22 — 2
25 — 1

34 — DI7-8
2 — 4
7 — 2
39 — 1

44 — DI8-8
13 — 4
19 — 2
56 — 1

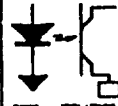
33 — DI9-8
1 — 4
6 — 2
36 — 1

43 — DI10-8
12 — 4
18 — 2
57 — 1

Speed
Select

3 to 8

Decoder



STOP
FWD

REV
REC
FAST

LOCAL

1 7/8 IPS
3 3/4 IPS
7 1/2 IPS
15 IPS
30 IPS
60 IPS
120 IPS
240 IPS

REMOTE COMMON

STOP
FWD

REV
REC
FAST

+5 STNG

1K

TSE
SEARCH
C0
FWD
ERROR

SYNC

HD2

HD1

TD8

TD4

TD2

TD1

UD8

UD4

UD2

UD1

TH2

TH1

UH8

UH4

UH2

UH1

TM4

TM2

TM1

UM8

UM4

UM2

UM1

TS4

TS2

TS1

US8

US4

US2

US1

tec8

tec4

tec2

tec1

FM
TAPE

J6

4
5
6
3
8
1
10
2

16
9
18
28
28

33
24
35

44

T C O

J2

8X
8D
8C
8E
8W

A

B

C

D

E

F

H

J

K

L

M

N

P

R

S

T

U

V

W

X

Y

Z

AA

AB

AC

AD

AE

AF

AH

AJ

AK

AL

AM

AN

FM TAPE CONTROL INTERFACE

BETWEEN: HP 88623A BCD INTERFACE
HONEYWELL 96 FM TAPE REC.
DATUM 9310 TIME CODE GEN.
Task 36
9/23/88 S. CLUKEY VOPAR RESEARCH ASSOCIATES

APPENDIX E. NEFF 130 Amplifier Interface

28 Dec 88

To: G. S. Jones
From: S. J. Clukey
Subject: INTERFACE CABLE - GAIN STATUS

The flow diagnostics test to be run in the 8'TPT to support choke evaluation testing will utilize a computerized data acquisition system known as DDAS. However, the gain settings of the amplifiers used to preprocess the Kulite data and the hot film transducers is not currently instrumented. These settings are essential for accurate data processing.

The NEFF 130-300 series of amplifier have a BCD output which indicates the gain setting of the amplifier. To read the gain code from this output with an existing computer and interface module, an interface cable with appropriate terminations is necessary. The NEFF amplifiers and related interface racks will be borrowed from others for the 8'TPT test.

The NEFF 130 amplifiers will be modified by lifting pin 13 of integrated circuit IC801. Bending this pin up, so it doesn't plug into the socket, continuously enables gain readout. The original design disabled gain readout when in manual gain selection mode.

The cable from the interface module is pre-terminated on the interface module end. The Neff 130 connectors have been procured. The junction box, barrier strips, strain relief devices, and cabling between the junction box and the NEFF 130 connectors are available from NASA stock.

The sketches showing electrical and mechanical details are attached.

Steve Clukey

HP 98623A
BCD I/O
CARD

NEFF
130

P1

HP Cable
5061-4271



junction box

: : : :
(10ea)
: : : :

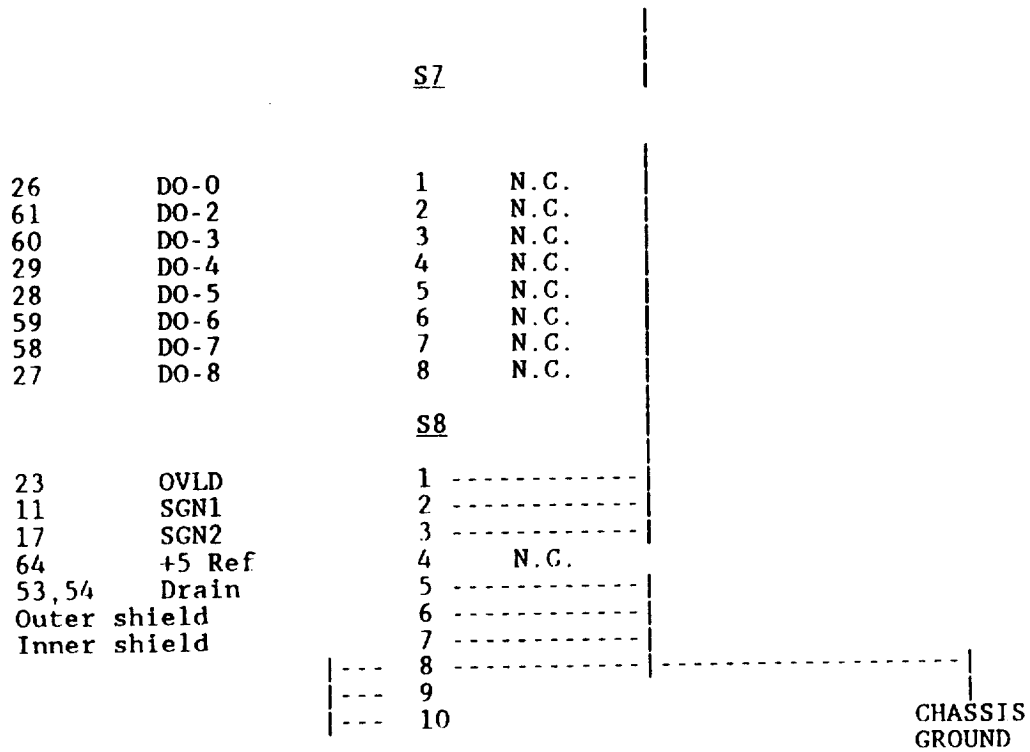
P8-01

P8-10

CABLE LAYOUT

BCD CARD	FUNCTION	JUNCTION BOX BARRIER STRIP	NEFF 130
<u>P1</u>		<u>S1</u>	[MS3106B-18-6P]
32	PRESET	1 N.C.	
62	CTLA	2 -----	
63	CTLB	3	
30	DFLGA	4 -----	
31	DFLGB	5	
		<u>S2</u>	<u>P8-01</u>
40	DI1-1	1 -----	A
8	DI1-2	2	B
3	DI1-4	3	C
35	DI1-8	4	D
		5	E
			<u>P8-02</u>
55	DI2-1	6 -----	A
20	DI2-2	7	B
14	DI2-4	8	C
46	DI2-8	9	D
		10	E
		<u>S3</u>	<u>P8-03</u>
41	DI3-1	1 -----	A
9	DI3-2	2	B
4	DI3-4	3	C
36	DI3-8	4	D
		5	E
			<u>P8-04</u>
24	DI4-1	6 -----	A
21	DI4-2	7	B
15	DI4-4	8	C
47	DI4-8	9	D
		10	E
		<u>S4</u>	<u>P8-05</u>
42	DI5-1	1 -----	A
10	DI5-2	2	B
5	DI5-4	3	C
37	DI5-8	4	D
		5	E

				<u>P8-06</u>
25	DI6-1	6	-----	A
22	DI6-2	7	-----	B
16	DI6-4	8	-----	C
45	DI6-8	9	-----	D
		10	-----	E
		<u>S5</u>		<u>P8-07</u>
39	DI7-1	1	-----	A
7	DI7-2	2	-----	B
2	DI7-4	3	-----	C
34	DI7-8	4	-----	D
		5	-----	E
				<u>P8-08</u>
56	DI8-1	6	-----	A
19	DI8-2	7	-----	B
13	DI8-4	8	-----	C
44	DI8-8	9	-----	D
		10	-----	E
		<u>S6</u>		<u>P8-09</u>
38	DI9-1	1	-----	A
6	DI9-2	2	-----	B
1	DI9-4	3	-----	C
33	DI9-8	4	-----	D
		5	-----	E
				<u>P8-10</u>
57	DI10-1	6	-----	A
18	DI10-2	7	-----	B
12	DI10-4	8	-----	C
43	DI10-8	9	-----	D
		10	-----	E
		<u>S1</u>		
48	Logic Ground	6	-----	
49	Logic Ground	7	-----	
50	Logic Ground	8	-----	
51	Logic Ground	9	-----	
52	Logic Ground	10	-----	



ACQUIRE	Operations Manual	Data Laboratories
ACQUIRE	Refence Maual	Data Laboratories
Multitrap	Operations Manual	Data Laboratories
Multitrap	Technical Manual	Data Laboratories
Multitrap Timebase	Technical Manual	Data Laboratories
Multitrap 1 chan digitizer	Technical Manual	Data Laboratories
Multitrap 4 chan digitizer	Technical Manual	Data Laboratories
BASIC	Operating System	Hewlett Packard
Precision Filter		Precision Filter
Array Processor		Analogic
	Technical Manual	
FM Tape		Honeywell
	Operation	
	Technical	
	Maintenance	
NEFF 130		NEFF
Time Code Generator		Systron Donner
BCD card		Hewlett Packard

APPENDIX F. Supporting and Related Documentation

< blank page >

INDEX

[...], 37
(...), 37
ACQUIRE, 11, 42, 43, 76, 77
ACQUIRE initialization, 33
ACQUIRE installation, 20
ACQUIRE key, 63
acquisition sequence, 25
action, 33
Aggregate bandwidth, 30
Amplifier and Filter Subsystem, 5
Amplifier and filter control software, 13, 48
AP CONST ARITH, 83
AP CONST K, 84
AP EVALUATE, 85
AP EVALUATE MORE, 86
AP INIT, 87
AP K*MEM, 90
AP K+MEM, 88
AP K-MEM, 89
AP K/MEM, 91
AP K:MEM, 92
AP MANIPULATE, 98
AP MATRIX, 99
AP MAX, 100
AP MEAN, 102
AP MEM A*MEM B, 95
AP MEM A-MEM B, 94
AP MEM A/MEM B, 96
AP MEM A:MEM B, 97
AP MEM ARITH, 103
AP MENU, 104
AP MIN, 101
AP REMOVE MEAN, 105
AP RMS, 106
AP SDEV, 107
AP SMOOTH, 108
AP SUM OF SQ, 109
AP TF FROM CPU, 111
AP TF MAP CHAN, 112
AP TF TO CPU, 113
AP TRANSFER, 110
APMEM A+MEM B, 93
APPLICATION, 38
Array Processor, 10
array processor integration, 16
AUTOSEQ, 24
AUTOST, 23
Bandwidth, 30
BASIC Operating System, 11
BEEP, 114, 231
Binary switches, 35
bottleneck, 30
Buffering, 30, 31
CALC NUSSELT, 115
CALC VEL etc, 116
Calculating fluctuations, 46
Calibration of wires, 39
calibration sequence, 24

- CAT GROUP, 118
- CATALOG, 117
- CODE TO GAINS, 119
- COEF FILENAME, 120
- Coefficient calculations, 44
- Coefficient Files, 41
- COMPUTE COEFS, 121
- COMPUTE R etc, 122
- COMPUTE SENS, 123
- Computer link, 6
- Configuration, 17
- Configuration files, 34
- CURRENT OBS, 124
- CURRENT PROBE, 125
- Data Laboratories, Ltd., 76, 77
- Data packet transfers, 15
- data flow, 28
- data management - logfile, 13
- data packet, 52, 53, 74, 75
- default test scenario, 21
- Digital voltmeter and scanner control, 15
- DIGITIZE ENABLE, 55, 126
- Disk, 9
- Display, 9
- DOS, 26
- DSP - Array Processor Integration, 61
- dynamic data, 42
- EDIT COEFS, 127
- END TIME, 128
- ENDING OBS, 129
- ENDING PROBE, 130
- ENHANCEMENTS, 36
- enhancements - sequence programs, 36
- ENTER COEFS, 131
- Execcmd, 34
- Execitem, 34
- Execstack, 34
- FILE COPY, 132
- FILE GROUP, 133
- FILE TRANSFERS, 134
- File conversion to DOS, 26
- File maintenance, 13
- file structure, 18
- file transfer to DOS, 26
- fluctuating data, 42
- fluctuating quantities, 29
- FM AUTOPLAYBACK, 51, 135
- FM AUTORECORD, 50, 136
- FM FORWARD, 137
- FM MOTION, 138
- FM POSITION, 50, 51, 139
- FM RECORD, 140
- FM RECORD TIME, 50, 141
- FM REVERSE, 142
- FM REWIND, 143
- FM SETUP, 144
- FM STOP, 145
- FM TAPE, 146
- FM TAPE SPEED, 147
- FM tape control, 14, 50
- FM Tape/Time Code Generator Interface, 8, 30
- FM TCG TIME, 148
- Format - file name (computed values), 77

Format - file name (fluctuating voltage), 76
FROM DISK, 149
function, 33, 81
Future enhancements, 60
General Purpose Interface Bus, 6
GET COEF, 150
GET MEAN DATA, 151
GET PF GAINS, 152
Get_packet, 8
Getting Started, 23
GPIB, 6, 7
grouping amplifier/filters, 49
HARDWARE DESCRIPTION, 5
hardware key, 63
HOTWIRE CALC, 154
HOTWIRE MENU, 153
hotwire anemometry, 28
HP-IB key, 63
HP-UX, 57
IDENT, 155
INITIAL OBS, 156
INITIAL PROBE, 157
initialization sequence, 24
Intr8, 53
key, 63
LOAD COEFS, 158
LOAD LOGFILE, 159
LOG DATA, 160
LOG DATA POINT, 161
LOG FILENAME, 162
log files, 40
LOGFILE TO PC, 54, 163
logfiles, 40
Mainloop, 33
MARK BAD DATA, 164
MARK CAL RAND, 165
MARK CAL SINE, 166
MARK CAL ZERO, 167
MARK GOOD DATA, 168
MARK IGNORE, 169
MARK OBS, 170
MATRIX CHECK, 171
mean quantities, 28
MOD7, 15
Moddl6000, 61
MODGEN, 76, 77
modifying the test scenario, 22
MODSEQ, 11, 37
MODUSR1, 12
MODUSR2, 8, 12, 13, 45, 76, 77
MODUSR3, 13
MODUSR4, 12, 14
MODUSR5, 14
MODUSR6, 15, 52
MODUSR8, 15
MOVE GROUP, 172
MULTITRAP, 43
Multitrap, 6
NEFF, 173
NEFF 130 AMPLIFIER Interface, 8, 32
NEFF 130 gain status, 14, 52
NEFF GAINS, 174
one button recording, 38

OPERATION, 17
optical disk, 17, 18, 22
Oswego, 26
P1, 175
P1 vars, 41
P2, 176
P3, 177
P3 vars, 41
P4, 178
PEN, 179
Peripherals, 9
PF CONTROL, 180
PF GAINS, 181
PF INIT, 182
PF SET PARAMS, 183
PF STATUS, 184
PICTURE LOG E, 185
PLAYBACK, 186
PLOT EJECT, 187
PLOT KING, 188
PLOT King, 188
PLOT LOG E, 189
PLOT UTILITIES, 190
plot setup, 35
Plotter, 9
PPRINT, 37
pre-calibrated, 21
PREC FILTERS, 191
PRINT, 37
PRINT DATA, 192
PRINT LAST OBS, 193
PRINT LOGFILE, 194
PRINT SENS, 195
PRINT TAPE LOG, 196
Printer, 10
PROBE A, 197
PROBE B, 198
PROBE C, 199
PROBE D, 200
PROCESS UTILITIES, 201
PROGRAM DEVELOPMENT, 56
PSHOW, 37
PURGE, 202
PURGE FILE, 203
PURGE GROUP, 204
RAW, 205
READY, 53, 206
REALTIME, 207
Remake Probe, 208
Reorder Obs, 209
REWRITE LOGFILE, 210
Routine, 33
S CALIB, 21, 24
S INIT, 21, 24
S RUN, 21
S TAPECAL, 25
SAMPLES TO AVG, 211
SCALAR PLOTS, 212
Scalar plots, 12, 54
Scanner/voltmeter control, 52
SDS, 52
SELECTOR, 213
SEND DATA, 214

SEND END, 215
SEND START, 216
SENS ALG, 217
Sensitivity calculations, 45
SEQ REMOTE GO, 55, 219
Sequence programs, 20, 22, 36
sequence functions, 11
SET F FILE NAMES, 77, 220
SET FILE NAMES, 76, 221
SET TIME, 222
setup - hardware, 17
SHOW, 37
SHOW " ", 37
SHOW DATA, 223
SHOW P1, 224
SHOW P2, 225
SHOW P3, 226
SHOW P4, 227
SHOW RAW, 228
SIMULT EQ, 229
software structure - ACQUIRE, 33
SOUND ALL DONE, 55, 230
SOUND GOT IT, 55
Spectral Data System, 52
START TIME, 232
START TIME FOUND, 233
Statistical Library, 41, 44
STOP TIME FOUND, 234
STORE COEFS, 236
STORE DATA ASCII, 235, 237
SUB, 33
SUMMARY, 58
Symbols, 2
SYSTEM DESCRIPTION, 5
System control, 13
System setup - software, 18
TAG PICTURE, 238
TAG PLOT, 239
tailoring the setup, 20
Tape, 9
tape calibration, 25
TCG, 50
Theory - hardware, 28
Theory - software, 33
time code generator/translator, 50
TO DISK, 240
TRIGGER, 53
trigger, 21
UNIX, 57
UTILITY, 54, 241
Utility functions, 12
voltmeter control, 52
WAIT !, 36, 51
WAIT *, 36, 51



Report Documentation Page

1. Report No. NASA CR-182069	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle A Real Time Dynamic Data Acquisition and Processing System for Velocity, Density, and Total Temperature Fluctuation Measurements		5. Report Date February 1991	
		6. Performing Organization Code	
7. Author(s) Steven J. Clukey		8. Performing Organization Report No.	
		10. Work Unit No. 505-60-21-04	
9. Performing Organization Name and Address ViGYAN, Inc. 30 Research Drive Hampton, VA 23666-1325		11. Contract or Grant No. NAS1-18585	
		13. Type of Report and Period Covered Contractor Report	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225		14. Sponsoring Agency Code	
		15. Supplementary Notes NASA Langley Research Center Technical Monitor: Leonard M. Weinstein	
16. Abstract <p>This report describes the real time Dynamic Data Acquisition and Processing System (DDAPS) which provides the capability for the simultaneous measurement of velocity, density, and total temperature fluctuations. The system of hardware and software is described in context of the wind tunnel environment.</p> <p>The DDAPS replaces both a recording mechanism and a separate data processing system. DDAPS receives input from hot wire anemometers. Amplifiers and filters condition the signals with computer-controlled modules. The analog signals are simultaneously digitized and digitally recorded on disk. Automatic acquisition collects necessary calibration and environment data. Hot wire sensitivities are generated and applied to the hot wire data to compute fluctuations. The presentation of the raw and processed data is accomplished on demand.</p> <p>This paper describes the interface to DDAPS and the internal mechanisms of DDAPS. A summary of operations relevant to the use of the DDAPS is also provided. This report supercedes CR 181758.</p>			
17. Key Words (Suggested by Author(s)) hot wire data acquisition anemometry three-wire dynamic data processing fluctuating measurement		18. Distribution Statement Unclassified - Unlimited Subject Category 03	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 286	22. Price A13